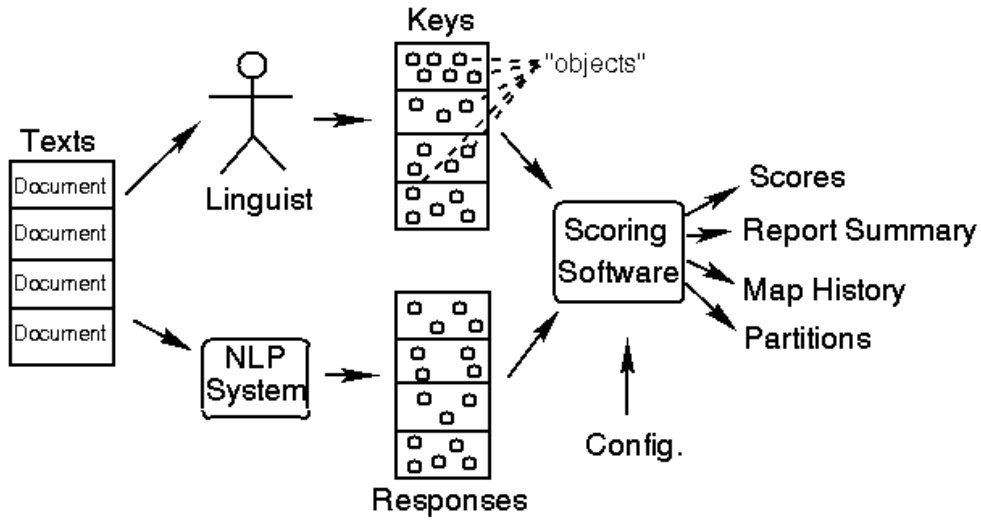
The Message Understanding Conference Scoring Software User's Manual

Introduction



The "MUC" Evaluation

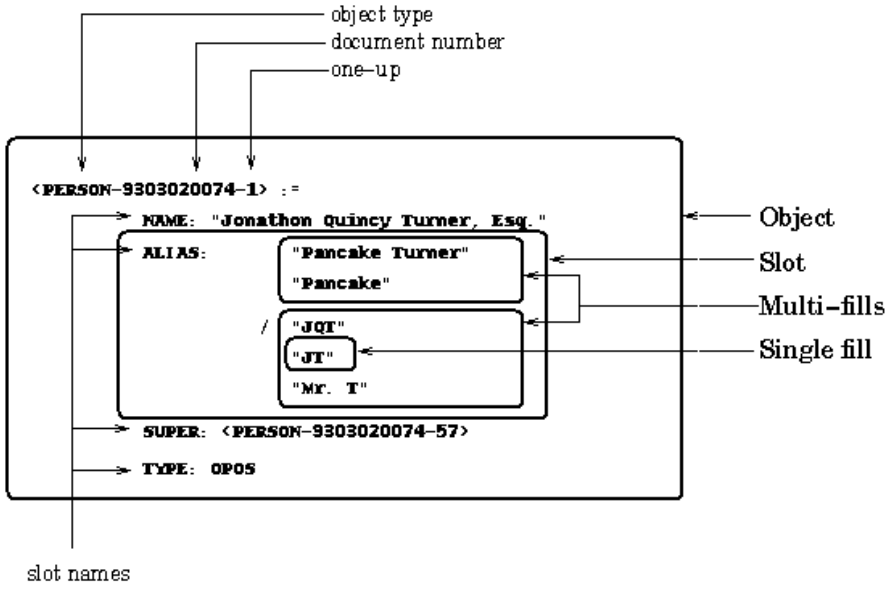
The Message Understanding Conferences (MUCs) are gatherings of researchers in computational linguistics. All participants in the conference develop software systems which perform natural language understanding *tasks* defined by the conference committee. The systems are evaluated based on how their output compares with the output of human linguists. The MUC scoring software is used in that comparison.

(This manual will briefly describe the MUC tasks from the standpoint of how the scoring software is used. For detailed, up-to-date descriptions of the the tasks, please refer to the various task definition documents for the conference.)

All tasks involve reading in a set of *documents* from one computer file. The documents contain text from periodicals or some other source of natural language, so the input file is called a *texts* file. The texts file is analyzed, and a set of *objects* is produced and printed to a single output file. For each document in the input file there are, in general, several objects produced. The format of the objects varies with the task, although the scorer uses the same internal representation of objects for all tasks.

When the scorer is run, it reads in an output file produced by a human, containing *keys*, and an output file produced by a software system, containing *responses*. The scorer *aligns* objects in the key file with objects in the response file. It then calculates various scores based on how well the responses agree with the keys. At present, there are two ways that the scores are calculated. In four of the five tasks, scores are based on counting how many *fills* (which are substructures of the objects, and are described below) agree for each aligned object. In the coreference task, the objects in the key file are grouped into equivalence classes, as are the objects in the response file. Scores for coreference are based on how well the equivalence classes agree.

The Structure of Scorer Objects



Internally, all objects compared by the scorer have the same basic hierarchical structure. We'll start at the bottom of this hierarchy and work our way up to the objects themselves. The figure above shows the terms we will introduce in relation to an object from a fictitious information extraction task.

For scoring, *Single fills* are treated as "atoms." When objects are aligned, the objects' single fills are also aligned, and it is the results of the single fill alignments which are tallied up to get the final score (in all tasks but Coreference). There are three types of single fills:

Set fills
: Strings of characters which must match exactly (except for character case) to be considered "correct".

String fills
: Strings of characters which must match after some massaging (e.g., removing articles or repeated space or tab characters) to be considered "correct".

Pointer fills
: References to other top level objects.

A *multi-fill* is a group of single fills.

A set of multi-fills is what goes into a *slot*. Slots in key objects may hold more than one multi-fill. Slots in response objects may hold only one multi-fill. When a key slot has more than one multi-fill, each key multi-fill is an *alternative*. The response slot's lone multi-fill will be aligned with whichever multi-fill of the key that results in the best score for the alignment. The unaligned multi-fills from the key are then non-committal. A slot also has a *name*, which distinguishes it from other slots in the top-level object.

A top-level *object* consists of some identification information and a set of slots. The identification information consists of an id string for the document from which the object was extracted (the *document number*), the object's *type*, and a string which distinguishes that object from all other objects of that type in that document. This string is sometimes called the *one-up* string.

Input File Formats

Template Files

Template files are the files produced in the information extraction tasks (TE, TR, and ST).

Response File Format

In a template file, the objects produced are in the form of *records*. An example object from a response file is shown here:

```
<ORGANIZATION-9303020074-1> :=
    ORG_NAME: "Evergreen Information Technologies Inc."
    ORG_ALIAS: "Evergreen Information Technologies"
               "Evergreen"
```

```
            "Evergreen Information"
    ORG_LEADER: <PERSON-9303020074-57>
    ORG_TYPE: COMPANY
```

Each record consists of a *header* and a list of *slots*. The header is an identification string for the object, followed by the token ":=" on the same line. The header's identification string is enclosed in angle brackets, and consists of three pieces of information:

- Object type
- Document Number
- One-up Number

Each slot in the body of the record consists of a slot name, followed by a colon, and the slot's fills. Set fills and string fills may be enclosed in matching single or double quotes. The format of pointer fills is the same as that of the string which identifies an object in its header.

Key File Format

Here is an example of a record from a human-generated key file:

```
<ORGANIZATION-9303020074-1> :=
    ORG_NAME: "Evergreen Information Technologies Inc."
    ORG_TYPE: /COMPANY
    ORG_ALIAS:  "Evergreen Information Technologies"
                "Evergreen"
                "Evergreen Information"
               /"Evergreen"
                "Evergreen Information"
    ORG_LEADER: <PERSON-9303020074-57>
    OBJ_STATUS: OPTIONAL
```

Key objects differ from response objects in a few respects:

- a slot may be marked "optional" by placing a slash character ("/") before the very first fill of the object. If the response object includes the optional slot, then the response fill and object fill are compared like any other fills. If the response object doesn't have the optional slot, no points are scored against it.
- a slot may contain "alternative" fills, separated by a slash character as the first non-blank character on a line. The response fill is matched with whichever of the alternatives gives the best "f"-score for that fill.
- the entire object may be marked optional, by including the "status" slot, with a fill of "optional". If an optional object is aligned with a response object, it is scored like any other object. But if no response object aligns with the optional object, no points are scored against the response.

Template File Caveats

The information extraction task descriptions often include a *BNF* which describes the different types of objects in the task. The scorer makes some some further assumptions about the format of template files which are not specified in the BNF's:

- all objects from a document should be grouped in one place in the template file.
- an object's header should be on its own line.
- if a line has a slot name, the name should be the first non-blank token on the line.
- there should be only one fill per line.
- a line containing a fill may have "link information" at the end of the line:

  ```
  SLOT_NAME: "a slot fill"  ##392#404#textsfilename
  ```

  This is a pair of pound signs ("##") followed by the "start offset" of the fill, then a single pound sign followed by the "end offset" of the fill, then another single pound sign, followed by the name of the texts file. None of the offset information is used in scoring, but it may be used in later versions of the scorer to highlight portions of the texts file. At present the scorer reads the start offset and end offset, but ignores the name of the texts file. The texts file name should not contain any pound signs.
- comments may be inserted into the template files on lines that have a pound sign or a semicolon as the very first character on a line.

SGML Task Files

The coreference and named entity tasks involve adding Standard Generalized Markup Language (SGML) to the the texts file to create the key and response files.

The Scoring Software's View of SGML

SGML is a very flexible and powerful language for adding structure to computer documents. The MUC scoring software recognizes a subset of SGML when it scores the coreference and named entity tasks. This discussion is a (very) simplified description of SGML.

An SGML *tag* is a character string inserted into a text file. Tags usually come in pairs, consisting of an *open tag* and a *close tag*. A pair of tags enclose a section of the text. For example, here is a piece of text, then the same text with some SGML tags added.

```
Be glad you don't work
On the Bungle-bung bridge,
That they're building
Across Boober Bay at Bum Ridge.
```

```
<ADVICE>
Be glad you don't work
On the <STRUCTURE>Bungle-bung bridge</STRUCTURE>,
That they're building
Across <BODY TYPE="WATER">Boober Bay</BODY> at <LOC>Bum Ridge</LOC>.
</ADVICE>
```

Open tags start with an open angle bracket, and are followed immediately by the *generic identifier* for that type of tag. Next come a sequence of *attribute* definitions for that type of tag. The end of an open tag is the close angle bracket. Close tags start with an open angle bracket, then a slash and the same generic identifier as close tag. Close tags don't have attributes.

In the above example, the three tag pairs have generic identifiers ADVICE, STRUCTURE, BODY, and LOC. Only the BODY tag has an attribute, named TYPE, with a value of WATER.

Conversion of SGML tags to MUC objects

In all MUC tasks, the texts file already has some SGML tags. In the coreference and named entity tasks, the annotators and systems add more tags to the texts to create the keys and responses. The scoring software converts the tags (together with the text they enclose) into objects which have the same internal structure as the objects for the information extraction tasks.

For example, here's some text marked up with TIMEX tags, which were part of the MUC6 named entity task.

```
<TIMEX TYPE="DATE" ALT="fiscal 1994">the first six months of fiscal 1994</TIMEX>
```

The scorer would convert the text into an object which in a template file would look like this:

```
<TIMEX-DOCNUM1-1> :=
    TEXT:  "the first six months of fiscal 1994"
          /"fiscal 1994"
    TYPE:  DATE
```

SGML task caveats

In the coreference and named entity tasks, there are some things to be careful of when you are preparing keys or responses. One thing is to not delete or insert any characters outside of the SGML tags. Doing this almost always confuses the scoring software and lowers the score. To see if you've changed anything you shouldn't have, you can use the unix "sed" command, or something similar, as in this example with the coreference tags:

```
unix% sed 's/<COREF[^>]*>//g' rsp | sed 's/<\/COREF[^>]*>//g' >rsp.notags
unix% diff texts rsp.notags
```

The `sed` command above removes the COREF tags from the responses file (named `rsp`), and then compares what's left to the original texts file (named `texts`). The `diff` command will then show what part of the original texts file has been changed.

Output File Formats

The MUC scoring software prints several reports to show how the key and response compared. There is a score report, which only shows "the numbers." There's also report summary, which shows in more detail how the key and response objects were aligned. For the coreference task, there is a "partitions" file, which shows how the key and response equivalence classes compared. And there is a "map history" file, which gives a detailed, if not very readable, description of how the objects were aligned.

Report Summary Files

The "report summary" files show how the fills and objects of the keys and responses align. There are three types of report summary files: one for the coreference task, one for the named entity task, and one for the information extraction tasks.

Coreference Report Summaries

Here's a section of a report summary file from the coreference task:

```
Document 930620083
COR "Clinton" "Clinton"
COR "Clinton" "Clinton"
COR "the White House" "White House"
COR "The current briefing room" "The current briefing room"
MIS "allies of the securities exchanges" ""
MIS "securities" ""
MIS "Clinton transition officials" ""
MIS "government" ""
MIS "the committee" ""
SPU "" "Kitty Higgins"
SPU "" "an aide"
SPU "" "Michigan"
OPT "the Clinton camp" ""
OPT "the government" ""
OPT "briefing" ""
```

A coreference report summary shows how the COREF objects were aligned by the scorer. Each line has three fields. The first field is a three letter abbreviation telling how a pair of objects are aligned. The abbeviations are:

COR
: Correct. The key and response objects agree.

MIS
: Missing. There was a key object but no response object.

SPU
: Spurious. There was a response object but no key object.

OPT
: Optional. There was a key object but no response object, but the key object was marked "optional".

The second field gives the text from the key object (if any), and the third field gives the text from the reponse object.

Named Entity Report Summaries

Here's a section of a report summary from the Named Entity task:

```
-------------------------------------------------------------------------------------------------------
Document 930620083
TAG        TYPE TEXT KEY_TYPE     RSP_TYPE     KEY_TEXT                                    RSP_TEXT
-------------------------------------------------------------------------------------------------------
ENAMEX     cor  cor  PERSON       PERSON       "Consuela Washington"                       "Consuela Washington"
ENAMEX     cor  inc  PERSON       PERSON       "John Dingell"                              "Washington"
ENAMEX     cor  inc  PERSON       PERSON       "Carter"                                    "Tim Wirth"
TIMEX      cor  cor  DATE         DATE         "01/19/93"                                  "01/19/93"
ENAMEX     mis  mis  PERSON                    "Washington"                                ""
ENAMEX     spu  spu               ORGANIZATION ""                                          "Exchange"
ENAMEX     spu  spu               ORGANIZATION ""                                          "Old Executive Office"
```

The named entity report summary file gives a one-line-per-object-pair description of how the objects were aligned. Each line has seven fields. The first is the generic identifier of the tag which defines the object. The second and third contain three-letter abbreviations for how the key and response objects or fills compared. The abbreviations are:

cor
: Correct. The key and response fills agree.

inc
: Incorrect. The key and response fills disagree.

mis
: Missing. There was a key fill but no response fill.

spu
: Spurious. There was a response fill but no key fill.

opt
: Optional. There was a key object but no response object, but the key object was marked "optional". The key object's fills are also counted as "optional".

The fourth and fifth fields are the key and response TYPE fills, if there are any. The sixth and seventh fields are the key and response TEXT fields. If the key contained more than one TEXT fill (through use of the ALT attribute), the one that was aligned with the response fill is the one shown.

If you are interested in seeing all alternatives, you can specify that you want to use the information-extraction-style report summary files. Just include the line

```
:use_IE_report_summary  yes
```

somewhere in the configuration file.

Information Extraction Report Summaries

This is a portion of a information extraction task (TE, TR, or ST) report summary file:

```
COR |                  | <PERSON-9301060123-8>                    | <PERSON-9301060123-8>
cor | PER_NAME:        |      Joe Roth                            |      JOE ROTH
cor | PER_ALIAS:       |      Roth                                |      ROTH
cor | PER_TITLE:       |      Mr.                                 |      MR.
COR |                  | <PERSON-9301060123-2>                    | <PERSON-9301060123-2>
inc | PER_NAME:        |      Rupert Murdoch                      |      MURDOCH
cor | PER_TITLE:       |      Mr.                                 |      MR.
mis | PER_ALIAS:       |      Murdoch                             |
SPU |                  |                                          | <PERSON-9301060123-5>
spu | PER_NAME:        |                                          |      SMITH BARNEY
SPU |                  |                                          | <PERSON-9301060123-12>
spu | PER_NAME:        |                                          |      RUPERT
COR |                  | <ORGANIZATION-9301130133-1>              | <ORGANIZATION-9301130133-1>
cor | ORG_NAME:        |      EMI Records Group                   |      EMI RECORDS GROUP
cor | ORG_TYPE:        |      COMPANY                             |      COMPANY
mis | ORG_ALIAS:       |      EMI Records                         |
mis | ORG_DESCRIPTOR:  |      a unit of London's Thorn EMI PLC    |
COR |                  | <ORGANIZATION-9301130133-2>              | <ORGANIZATION-9301130133-2>
uns | COMMENT:         |      the alias 'EMI' is here assumed...  |
cor | ORG_NAME:        |      Thorn EMI PLC                       |      THORN EMI PLC.
cor | ORG_TYPE:        |      COMPANY                             |      COMPANY
mis | ORG_ALIAS:       |      EMI                                 |
mis | ORG_LOCALE:      |      London CITY                         |
mis | ORG_COUNTRY:     |      United Kingdom                      |
```

The information extraction report summaries files have four columns. The first column shows the result of the pairing on that line. Upper case values are for object comparisons, and lower case values are for single fill comparisons. Possible values are:

cor
: Correct. The key and response agree.

inc
: Incorrect. The key and response disagree.

mis
: Missing. There was a key but no response.

spu
: Spurious. There was a response but no key.

opt
: Optional. There was a key but no response, and the key object or slot was marked optional.

uns
: Unscored. The object or slot isn't scored.

rem
: Removed. This is for pointers to optional objects. If a key pointer points to an optional key object that was not aligned with any response object, the fill is "removed," and doesn't count toward the score.

The second column shows the name of the slots for the key and response object for the line (and the lines following if there are multiple fills in the slot).

The third and fourth columns show the key and response object records, respectively.

Coreference "Partition" Files

For the coreference task, there is an extra report generated, which shows the COREF objects' equivalence classes, and how they are partitioned by the comparison between keys and responses. Key equivalence classes are surrounded by star characters (*****), and response equivalence classes by equal signs (=====).

Here is a portion of a partition file that gives one key equivalence class from a MUC 6 document.

```
**************************************************

C 88 116 1 NULL 108 "Washington, an Exchange Ally,"

C 581 609 4 0 39 "Ms. Washington, 44 years old,"
C 741 754 8 4 40 "Ms. Washington"
C 828 830 9 8 43 "her"
C 916 918 12 9 42 "She"
C 961 974 15 12 41 "Ms. Washington"

C 1124 1171 20 15 48 "A graduate of Harvard Law School, Ms. Washington"
C 1257 1259 22 20 49 "She"

M 376 454 0 1 "Consuela Washington, an expert in securities laws,"

**************************************************
```

Each line containing COREF objects begins with a "C" or an "M", for "correct" or "missing." Correct objects' lines have, in order from left to right,

- the start offset of the noun phrase in the texts file.
- the end offset of the noun phrase in the texts file.
- the ID of the key COREF object.
- the ID of the key COREF object to which this object points (or "NULL" if the object has no REF attribute).
- the ID of the response COREF object aligned with the key coref object.
- the noun phrase that was marked up to create the object.

In the "missing" objects' lines, the fields are the same except the response object's ID is, of course, missing. Note that there are blank lines between some of the COREF object lines. These show the partitions of the key equivalence class by the response. While the key ties together every noun phrase between the stars, the response doesn't, so there are "breaks" in the equivalence class. These breaks are what are counted to get the recall error. The precision error is got from the response equivalence classes in a symmetric manner.

Map History Files

The "map history" output file is meant primarily for other computer programs to read. It consists of one large Tcl-style list. Each element of this list is itself a list which corresponds to one "document" from the keys and/or responses file. The document lists also contain lists, and this nesting of lists continues on down to the single fill level. Lists in the hierarchy consist of attribute name/attribute value pairs. Attribute names start with a hyphen.

The Hierarchy of Map History Lists

In hierarchy order, the attributes are:

Document level
- -docnum
- -doctallies
- -class_pairs

Class pair level
- -class_name
- -class_tallies
- -obj_pairs

Object Pair level
- -obj_pair_status
- -obj_pair_tallies
- -key_obj_id
- -key_obj_optional
- -key_obj_rep_id
- -key_obj_start_offset
- -key_obj_end_offset
- -rsp_obj_id
- -rsp_obj_optional
- -rsp_obj_rep_id
- -rsp_obj_start_offset
- -rsp_obj_end_offset
- -doc_section
- -slot_pairs

Slot pair level
- -slot_name
- -slot_tallies
- -key_slot_optional
- -rsp_slot_optional
- -multi-fill_pairs

Multi-fill pair level
- -multi_fill_tallies
- -single_fill_pairs

Single fill pair level
- -single_fill_pair_status
- -single_fill_pair_tallies
- -key_single_fill
- -rsp_single_fill

Single fill substructure level

- -type
- -fill
- -clean_fill
- -start_offset
- -end_offset

Description of Map History Attributes

Attribute values are strings, lists, integers, or nonexistent if the attribute's presence alone implies something. At present, the attributes are:

class_name
: the name of a object type, e.g. "ENAMEX".

class_pairs
: a list describing how the groups of objects of the same type were aligned.

class_tallies
: single-fill tallies for all objects of one type in one document.

clean_fill
: How a single string fill looks when it is compared to another fill. Leading and trailing whitespace has been trimmed, certain substrings have been removed, and all intertoken whitespaces are changed to single space characters. For example, the string "a corporation that manages the Seaport" would be changed to "that manages the seaport" (depending on how the scorer is configured), because the premodifier "a" and the corporate designator "corporation" are both removed, and all characters are made lowercase.

docnum
: the string identifying the document in the texts file

doctallies
: the totals of the (in order) possible, actual, correct, partial, incorrect, missing, spurious, and noncommittal single-fill "tallies" for the entire document.

doc_section
: in the SGML tasks (named entity and coreference), the name SGML tags which enclose the object in the texts document, e.g. "HEADLINE" or "TEXT".

fill
: the fill as it appeared in the key or response file (with one exception: in the coreference task's REF fill, this is how the REF attribute would look if it were written as a template object pointer).

key_obj_id
: The identification string of the key object of the pair.

key_obj_optional
: Whether the object in the key is marked optional. This attribute has no value following it in the list; its presence alone means the key was marked optional.

key_obj_rep_id
: Almost always the same as the key_obj_id. In the scenario template task of past MUC's, there have been objects in the key that are "identical". All objects are put in equivalence classes (different from the equivalence classes of the coreference task), so that pointers to any object in an equivalence class are still counted correct, even though they don't point to exactly the same object.

key_obj_end_offset
: in the SGML tasks (named entity and coreference), the position in the texts file, measured from the beginning of the file, where the close tag for the object is.

key_obj_start_offset
: in the SGML tasks (named entity and coreference), the position in the texts file, measured from the beginning of the file, where the open tag for the object is.

key_single_fill
: a list describing the single fill from the key.

key_slot_optional
: whether the slot was marked optional in the key. This attribute has no value associated with it. If the attribute name is there, it means the slot was marked optional.

multi_fill_pairs
: the list describing how the key slot fill alternatives were aligned with the response alternatives. (When scoring system responses, there should be only one response alternative. For interannotator comparisons, both key and response may have many alternatives.)

multi_fill_tallies
: the tallies for the single fills in this pairing of alternatives (see multi_fill_pairs).

obj_pair_status
: How the objects of a pair compared at the object-level; correct, incorrect, etc.

obj_pair_tallies
: the tallies for the single fills in this pair of objects.

obj_pairs
: a list describing how objects of one type were aligned.

rsp_obj_id
: The identification string of the rsp object of the pair.

rsp_obj_optional
: Whether the object in the response is marked optional.

rsp_obj_rep_id
: Almost always the same as the rsp_obj_id. In the scenario template task of past MUC's, there have been objects in response that are identical. All objects are put in equivalence classes (different from the equivalence classes of the coreference task), so that pointers to any object in an equivalence class are still counted correct, even though they don't point to exactly the same object.

rsp_single_fill
: a list describing the single fill from the response.

single_fill_pair_status
: A three-character abbreviation for how the two single fills in a pair compared.

single_fill_pair_tallies
: Another way for writing the single_fill_pair_status, that is compatible with all other tallies up the hierarchy.

single_fill_pairs
: the list describing how one list of key single fills (possibly from many alternatives) was aligned with one list of response single fills.

slot_name
: the name of the slots which are paired here.

slot_pairs
: the list describing how the two objects' slots compared.

slot_tallies
: the tallies for the single fills in this slot's comparison.

type
: the type of the single fill (set fill, string fill, or pointer fill).

Score Files

Information Extraction Score Report

Figure shows one page from a *scores* file for the MUC-6 scenario template task. There is one page of scores for each document in the task, plus one page for the totals over all documents. Each page is divided into four sections. The first section shows the "text filtering" or "relevance" scores. These have to do with judging whether each document is even relevant to the scenario the NLP system should be looking for. The second section gives the object scores, which shows how the keys and response agree at the object level. The third section shows how well the keys and responses agree at the slot fill level. Only the slot scores determine the final scores, which are the last thing on a page.

The template element and template relation score reports are identical to the scenario template score reports, except that they have no text filtering section.

	POS	ACT	COR	PAR	INC	MIS	SPU	NON	REC	PRE	UND	OVG	SUB	ERR
TEXT FILTERI	100	100	86	0	14	0	0	0	86	86	0	0	14	14
OBJ SCORES														
template	0	0	0	0	0	0	0	100	0	0	0	0	0	0
succession_e	195	197	131	0	6	58	60	13	67	66	30	30	4	49
in_and_out	256	310	164	0	6	86	140	20	64	53	34	45	4	59
organization	110	72	51	0	1	58	20	10	46	71	53	28	2	61
person	130	138	90	0	5	35	43	7	69	65	27	31	5	48
SLOT SCORES														
template														
doc-nr	0	0	0	0	0	0	0	100	0	0	0	0	0	0
content	195	197	131	0	6	58	60	51	67	66	30	30	4	49
comment	0	0	0	0	0	0	0	15	0	0	0	0	0	0
succession_e														
success_org	193	128	69	0	26	98	33	15	36	54	51	26	27	69
post	195	170	65	0	51	79	54	35	33	38	41	32	44	74
in_and_out	251	191	55	0	25	171	111	26	22	29	68	58	31	85
vac_reason	195	197	63	0	74	58	60	36	32	32	30	30	54	75
comment	0	0	0	0	0	0	0	273	0	0	0	0	0	0
in_and_out														
io_person	254	304	126	0	39	89	139	22	50	41	35	46	24	68
new_status	256	304	138	0	32	86	134	20	54	45	34	44	19	65
on_the_job	256	310	107	0	63	86	140	73	42	35	34	45	37	73
other_org	168	5	3	0	2	163	0	47	2	60	97	0	40	98
rel_oth_org	172	5	3	0	2	167	0	34	2	60	97	0	40	98
comment	0	0	0	0	0	0	0	399	0	0	0	0	0	0

organization														
name	108	69	28	0	21	59	20	10	26	41	55	29	43	78
alias	65	42	12	0	4	49	26	16	18	29	75	62	25	87
descriptor	64	2	0	0	2	62	0	50	0	0	97	0	100	100
type	110	69	50	0	1	59	18	12	45	72	54	26	2	61
locale	41	7	4	0	3	34	0	8	10	57	83	0	43	90
country	41	7	6	0	1	34	0	5	15	86	83	0	14	85
comment	0	0	0	0	0	0	0	15	0	0	0	0	0	0
person														
name	130	138	82	0	13	35	43	7	63	59	27	31	14	53
alias	83	79	56	0	3	24	20	5	67	71	29	25	5	46
title	79	78	60	0	0	19	18	5	76	77	24	23	0	38
comment	0	0	0	0	0	0	0	1	0	0	0	0	0	0
ALL SLOTS	2856	2307	1058	0	368	1430	881	1280	37	46	50	38	26	72
									P&R		2P&R		P&2R	
F-MEASURES									40.98		43.78		38.53	

Named Entity Score Report

Here is a page from a score report for the named entity task:

	POS	ACT	COR	PAR	INC	MIS	SPU	NON	REC	PRE	UND	OVG	SUB	ERR
SUBTASK SCORES														
enamex														
organizatio	443	444	405	0	18	20	21	18	91	91	5	5	4	13
person	373	371	364	0	2	7	5	0	98	98	2	1	1	4
location	110	122	109	0	0	1	13	3	99	89	1	11	0	11
other	0	0	0	0	0	0	0	0	0	0	0	0	0	0
timex														
date	111	112	107	0	0	4	5	6	96	96	4	4	0	8
time	0	0	0	0	0	0	0	0	0	0	0	0	0	0
other	0	0	0	0	0	0	0	0	0	0	0	0	0	0
numex														
money	76	76	73	0	0	3	3	0	96	96	4	4	0	8
percent	17	25	17	0	0	0	8	0	100	68	0	32	0	32
other	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SECT SCORES														
Header	244	256	233	0	9	2	14	8	95	91	1	5	4	10
Body	2016	2044	1906	0	42	68	96	95	95	93	3	5	2	10
OBJ SCORES														
enamex	926	937	898	0	0	28	39	21	97	96	3	4	0	7
timex	111	112	107	0	0	4	5	6	96	96	4	4	0	8
numex	93	101	90	0	0	3	11	0	97	89	3	11	0	13
SLOT SCORES														
enamex														
type	926	937	878	0	20	28	39	21	95	94	3	4	2	9
text	926	937	876	0	22	28	39	21	95	93	3	4	2	9
status	0	0	0	0	0	0	0	38	0	0	0	0	0	0
alt	0	0	0	0	0	0	0	0	0	0	0	0	0	0
timex														
type	111	112	107	0	0	4	5	6	96	96	4	4	0	8
text	111	112	98	0	9	4	5	11	88	88	4	4	8	16
status	0	0	0	0	0	0	0	6	0	0	0	0	0	0
alt	0	0	0	0	0	0	0	0	0	0	0	0	0	0
numex														
type	93	101	90	0	0	3	11	0	97	89	3	11	0	13
text	93	101	90	0	0	3	11	0	97	89	3	11	0	13
status	0	0	0	0	0	0	0	0	0	0	0	0	0	0
alt	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ALL SLOTS	2260	2300	2139	0	51	70	110	103	95	93	3	5	2	10
									P&R		2P&R		P&2R	
F-MEASURES									93.82		93.32		94.31	

The report has several parts:

subtask scores
: Each named entity tag contains an attribute categorizing the marked-up text. This section shows how well the response did for each category.

section scores
: Each document is already marked up with SGML even before the keys and responses are made. This section summarizes how the response did for each "section" of the SGML document.

object scores
: Tallies at the object level. These tallies don't contribute to the final score at the bottom of the page.

slot scores
: Tallies at the slot level. It is the slot level tallies which are used to determine the final score.

Coreference Score Report

Here is a coreference task score report:

Key Rsp

```
Document No.      Cls   Cls          Recall                Precision       f

930620083          23    20      25 / 43     58.1        25 / 40     62.5   60.2
930620057           4     5      11 / 15     73.3        11 / 14     78.6   75.9
930560132          11     6      14 / 20     70.0        14 / 16     87.5   77.8
930380019          18    10      50 / 69     72.5        50 / 59     84.7   78.1
930350079           3     2       8 / 11     72.7         8 / 10     80.0   76.2
930220297          34    39     101 / 157    64.3       101 / 133    75.9   69.7
930220050           2     5       1 / 3      33.3         1 / 6      16.7   22.2
930090013           8     2       7 / 17     41.2         7 / 8      87.5   56.0
930050011          10    13      26 / 35     74.3        26 / 31     83.9   78.8
931290244          12     8      11 / 21     52.4        11 / 16     68.8   59.5
931250227          35    26      37 / 85     43.5        37 / 59     62.7   51.4
931110023           5     9      25 / 36     69.4        25 / 33     75.8   72.5
931090230           4     3       8 / 12     66.7         8 / 10     80.0   72.7
931020207          10    10      25 / 41     61.0        25 / 32     78.1   68.5
930900283          16    16      52 / 75     69.3        52 / 66     78.8   73.8
930860108           7     6      20 / 31     64.5        20 / 23     87.0   74.1
930710271           3     4       5 / 8      62.5         5 / 6      83.3   71.4
940460255           3     7      11 / 17     64.7        11 / 14     78.6   71.0
940430215           4     3       4 / 9      44.4         4 / 5      80.0   57.1
940430078           5     6      15 / 19     78.9        15 / 19     78.9   78.9
940410075          13    14      22 / 34     64.7        22 / 32     68.8   66.7
940370255          41    43      72 / 133    54.1        72 / 115    62.6   58.1
940280231          44    50      65 / 126    51.6        65 / 103    63.1   56.8
940270193          22    23      46 / 75     61.3        46 / 69     66.7   63.9
940260231          14    25     102 / 124    82.3       102 / 121    84.3   83.3
940190235          22    21      61 / 78     78.2        61 / 81     75.3   76.7
940120142          31    35     116 / 165    70.3       116 / 146    79.5   74.6
940090210          18    22      33 / 59     55.9        33 / 59     55.9   55.9
940080212           4     3       5 / 7      71.4         5 / 7      71.4   71.4
940050261           9     5      12 / 21     57.1        12 / 12    100.0   72.7

TOTALS:           435   441     990 / 1546   64.0%      990 / 1345   73.6%  68.5%
```

There is one line for each document in the corpus. From left to right, the fields of each line are:

1. the document number of the line's article.
2. The number equivalence classes of COREF objects in the key and response, respectively.
3. The recall score, as a fraction and as a percent.
4. The precision score, as a fraction and as a percent.
5. the f-score, if you give recall and precision equal weight.

Configuration File Formats

The scoring software has three configuration files, that you use to specify how the keys and responses are compared. The reason there are three files is partly historical and partly because parsing some of the configuration options differs a little. In future versions the three files will probably coalesce into one file.

Main Configuration File Format

You must specify the name of the main configuration file on the command line when you invoke the scorer. The configuration file tells the scorer how to compare the keys and responses. It consists of a list of *options*. Each option is specified by a colon (":") as the first character of a line, followed immediately (no spaces) by the name of the option. After some more spaces come the value or values of the option. Values are separated by spaces. Values which themselves contain spaces must be enclosed in single or double quotes. The current options are:

class_defs

The strings which declare the scorer objects' types and give their score report names and mapping order. This is a required entry in the configuration file. Each class_def string is a quadruple of tokens:

1. the name of the class
2. the name of the class that you want to appear in the score report
3. either "scored" or "unscored," depending on whether you want the object-level scoring to count this class of object.
   Note that this value doesn't affect whether the fills within the objects are scored. Slot-level scoring is specified in the slot_defs option, described below.
4. the *map threshold*. The f-score for each slot of an object is calculated and multiplied by that slot's *map weight*. The weighted f-scores are them summed, and if they exceed the object's map threshold, then the response and key objects are deemed similar enough to be aligned. For the past couple of MUC's, the threshold has been set to 0 and the map weights have be made really big, so that if the two objects agree in just one fill of one slot, they may be aligned.

Here is an example of the class_def option, for the named entity task:

```
:class_defs
       "enamex  enamex  scored  0"
       "numex   numex   scored  0"
       "timex   timex   scored  0"
```

The class def strings should be in the order that you want the classes of objects aligned. For the named entity, template element and coreference tasks, this order is unimportant. But for the template relation and scenario template tasks, the pointer fills are judged correct or incorrect based on whether or not the objects they point to are aligned. So the aligning should always start with objects that contain no pointer fills, and proceed to objects whose only pointer fills reference objects without pointer fills, etc. (See the section on how the TR and ST tasks are scored, below.)

content_name
: In the scenario template task, the name of the slot in the "template" object (see the template_name option below) which must have fills if the document is relevant to the ST task, and which must not have fills if the document is not relevant to the ST task. Default: "content".

corporate_designators
: A list of substrings which will be removed from string fills before they are compared. As the name implies, it's usually a list of strings like "corporation", "ltd", etc. Note that if you want to remove substrings that themselves have postmodifiers (see below), you must specify the substrings with postmodifiers changed to spaces, and all resulting spaces in the corporate designator string squashed into one. For instance, if you don't want the string "S.A. DE C.V." to affect stringfill comparisons, it should go into the configuration file as "S A DE C V", with one space between the "A" and the "DE". (But for the coreference task, you should *not* take out the postmodifiers.) Default: the empty list.

doc_section_groups
: Used in the Named Entity task, to group doc_section (see below) scores. In MET 2, the documents in the texts file have various SGML formats. For example, in some documents there is a HEADLINE tag, but in other documents, the tag is called HL. To get a score for all document sections which are the same semantically, but differ in their tags, you can "group" the similar tags, by putting, for example, "Headline HEADLINE HL" as one value for this option. The first token in a value string is what you want to call the group. The rest of the tokens are the name of doc sections. You must also specify the rest of the tokens in the "doc_sections" option described below. If this option is not in the configuration file, the doc_sections scores are used. If it's specified, the scorer only gives the tallies for the names given. Default: None; uses doc_sections instead.

doc_sections
: The names of the SGML sections that should be parsed for coreference or named entity objects, and which will be used to report "document section scores" in the named entity task. The default is this list of sections: <DOC>, <DATELINE>, <DD>, <HEADLINE>, and <TEXT>. Note that as long as the documents are enclosed by <DOC>, all of the objects will be parsed by default. Having tags that don't really occur in the documents won't hurt anything. If one section is nested in another section, it is the innermost section which will be reported for the score. For example, if there are HEADLINE's inside the TEXT, the objects will be considered to be inside the HEADLINE.

dump_map_history
: Whether or not to print the map history report. Default: "no". If anything else, the map history will be printed.

equatable_objects
: In the scenario template task, which objects may possibly be identical. In MUC 6, the "IN_AND_OUT" objects were like this. Default: no objects.

key_file
: The name of the keys file. Default: "keys".

map_history_file
: The name of the map history output file. Default: "map_history"

muc_base_directory
: A string that is prepended to the names of all filename options. This allows you to give the absolute pathname of all filenames without a lot of typing. Defaults to the empty string.

ne_subtask_names
: A list of strings, each with three tokens. The first token is the object type. The second token is the slot name. The third token is the fill value. The tallies for all fills of that value in that slot in that type of object will be reported in the NE subtask section of the score report. Default: the following strings:
  - "enamex type organization"
  - "enamex type person"
  - "enamex type location"
  - "enamex type other"
  - "timex type date"
  - "timex type time"
  - "timex type other"
  - "numex type money"
  - "numex type percent"
  - "numex type other"

optional_status_slot
: The name of the slot in *all* objects that you use to specify that an objects is optional, by putting the string "OPTIONAL" or "OPT" as the slot's only fill.

partition_file
: The name of the partition output file for the coreference task.

postmodifiers

A list of strings that are changed to spaces in stringfills before the stringfills are compared. Usually used so that punctuation marks don't affect the comparisons. Default: the empty list.

premodifiers
: A list of tokens that are removed from the beginning of stringfills before they are compared. Usually used so that the words "a," "an," and "the" don't affect the scoring.

report_field_separator
: A character string that is printed between the fields of the information extraction-style "report summary" files. The default is the vertical bar ("|").

report_summary_file
: The name of the report summary file. Default: "report_summary".

response_file
: The name of the responses file. Default: "responses".

score_report_file
: The name of the scores file. Default: "scores".

scoring_method
: One of either "key2response" or "key2key". "Key2response" is the default. Key2key is used for interannotator comparisons.

scoring_task
: One of "coreference", "named_entity", "template_element", "template_relation", and "scenario_template." There is no default for this option. It must specified.

sgml_ALT_slot
: In the named entity task, the name of the slot whose contents will be moved into the TEXT slot (see sgml_TEXT_slot below), as an alternative to the contents got from the text between the SGML tags.

sgml_DOCNUM_gid
: The name of the SGML tag which identifies the section which holds the document numbers. Default: DOCNO. Note that every document in the keys or responses file must have this section. The document number is simply every digit [0-9] in the specified document section.

sgml_DOC_gid
: The name of the SGML tags which enclose one entire document. Default: "DOC".

sgml_ID_slot
: In the coreference task, the name of the attribute of the tags for the task which give the unique identification string for the object. Default: "ID".

sgml_MIN_slot
: In the coreference task, the name of the attribute which holds the "head" of the noun phrases enclosed in the coreference tags. Default: "MIN".

sgml_REF_slot
: In the coreference task, the name of the attribute which holds the pointer some other "identical" object in the document. Default: "REF".

sgml_TEXT_slot
: In the named entity and coreference tasks, the name of the slot into which the text between the open and close tags goes. Default: "TEXT".

sgml_TYPE_slot
: In the named entity task, the name of the slot for the categorization subtask. Default: "TYPE".

sgml_alternative_separator
: In the named entity and coreference tasks, the character which separates alternatives within attribute values. Note that for the current tasks, this is only relevant for the keys. Default: the vertical bar character, ("|").

sgml_attribute_quote_char
: In the named entity and coreference tasks, if a tag's attribute value is a string which contains a double quote, the scorer's parser will become confused. This option contains the character which has been substituted for the double quote in the keys or responses file. (Again, in the current tasks, this will only affect how the keys are prepared, since the response don't have attributes that might contain quotes.) Default: the "star" character ("*").

slot_defs
: The list of slot definitions. This is a required entry in the configuration file. Each slot definition consists of six tokens:
  1. The name of the class of object to which the slot belongs.
  2. The name of the slot.
  3. The name of the slot that you want printed in the score report file.
  4. Either "scored" or "unscored," depending on whether you want the fills of this slot to be scored.
  5. The *map weight*. See the entry for the class_def option for an explanation of this number.
  6. The slot type; either "set", "string", or "pointer" (you may put anything here for a pointer slot. The scorer only looks to see that it isn't "set" or "string").

  Here's an example of the slot_defs option for the named entity class:

```
:slot_defs
          "enamex       text    text    scored          4       string"
          "enamex       type    type    scored          4       set"
          "enamex       status  status  unscored        4       set"
```

```
"enamex       alt     alt     unscored        4       string"

"timex        text    text    scored          4       string"
"timex        type    type    scored          4       set"
"timex        status  status  unscored        4       set"
"timex        alt     alt     unscored        4       string"

"numex        text    text    scored          4       string"
"numex        type    type    scored          4       set "
"numex        status  status  unscored        4       set"
"numex        alt     alt     unscored        4       string"
```

stringfill_correct_comparison
: one of "ORIG", "STRAIGHTENED", or "CLEAN". Which part of a pair of stringfills is compared to see if they match. If ORIG, the original stringfills are compared. If STRAIGHTENED, some massaging is performed: Whitespaces are trimmed before and after the fills, and all whitespaces between the tokens are turned into single spaces. If CLEAN, the premodifiers, postmodifiers, and corporate designators strings (see the option descriptions for these last three) are removed from the string. Default: CLEAN

stringfill_partial_comparison
: If the stringfills don't match correctly, the comparison used to see if partial credit is given for the match. See stringfill_correct_comparison for the possible values. In additions to the three values listed there, you may specifiy NONE (the default) if you want no partial credit given.

template_name
: The name of the "template" object used in the scenario template object. This object has a "content" slot (see the "content_name" option) whose filling or leaving empty determines whether document is relevant to the scenario. For scoring the text-filtering part of the task, only one one template object per document will be checked for content. Default "TEMPLATE".

use_IE_report_summary
: Defaults to "no". If anything else, the one-line-per-object report summaries used for the named entity and coreference tasks will be replaced with the template-object-record-style report summaries used in the Information Extraction tasks. (This option doesn't affect the TE, TR, or ST tasks).

Calculation of Scores

Template Element (TE) Scoring

The methods for scoring the Template Element, Template Relation, Scenario Template, and Named Entity tasks are very similar. From the standpoint of calculating scores, The template element (TE) task is the basic task of these four. This section will explain how TE is scored, and subsequent sections will tell how the NE, TR, and ST tasks can be seen as extensions to TE scoring.

Simply put, the final score for the four tasks is found by aligning the key objects with the response objects and then comparing the objects' single fills. Structures are aligned at each level of the object/slot/multi-fill/single-fill structure hierarchy. However, it is the single-fill alignments that we count to get the score.

The result of aligning one key single fill to one response single fill (or of leaving one key or response single fill unaligned) is called a *tally*. There are six kinds of tallies:

COR Correct
: the two single fills are considered identical.

INC Incorrect
: the two single fills are not identical.

PAR Partially Correct
: the two single fills are not identical, but partial credit should still be given.

MIS Missing
: a key object has no response object aligned with it.

SPU Spurious
: a response object has no key object aligned with it.

NON Noncommittal
: the alignment doesn't contribute anything to the scoring.

Given a set of tallies, there are several values calculated in the alignment and final scoring.

POS Possible
: The number of fills in the key which contribute to the final score.

```
POS = COR + INC + PAR + MIS
```

ACT Actual

The number of fills in the response.

$$ACT = COR + INC + PAR + SPU$$

REC Recall

a measure of how much of the key fills were produced in the response.

$$REC = \frac{COR + (0.5 * PAR)}{POS}$$

PRE Precision

a measure of how much of the response fills are actually in the key.

$$PRE = \frac{COR + (0.5 * PAR)}{ACT}$$

Intuitively, information extraction systems often sacrifice precision for recall, or vice versa. If a system is tuned to "catch everything" (good recall), it often catches more than it should (bad precision). And if it tries to be conservative (good precision), it tends to miss some information (bad recall). When evaluating responses, then, one has to be careful about comparing one response from a system tuned for high recall to another response from a system tuned for high precision. van Rijsbergen's F-measure is used to combine recall and precision measures into one measure. The formula for F is

$$F = \frac{((beta)^2 + 1.0) * P * R}{((beta)^2 * P) + R}$$

where beta is the relative weight of precision and recall.

The following measures are also calculated from the tallies, and are in the score report:

UND Undergeneration

$$UND = \frac{MIS}{POS}$$

OVG Overgeneration

$$OVG = \frac{SPU}{ACT}$$

SUB Substitution

$$SUB = \frac{INC + (0.5 * PAR)}{COR + INC + PAR}$$

ERR Error per response fill

$$ERR = \frac{INC + (0.5 * PAR) + SPU + MIS}{COR + INC + PAR + SPU + MIS}$$

When aligning two multi-fills, the scoring software pairs all single-fills of the multi-fills. For example, if the key multi-fill has three single-fills, and the response multi-fill has two multi-fills, then the scorer creates six pairs of single-fills. Each single-fill pair has an F-score associated with it. The scorer sorts these single-fill pairs by F-score in decreasing order. It then proceeds down the sorted list, picking out pairs of single-fills for which neither single-fill has been chosen yet, and adding them to the final alignment for that pair of multi-fills. Any key or response single fills left over (in our example, there would be a key single fill left) is tallied as missing or spurious.

A key slot is aligned with a response slot when the two slots have the same name. The lone multi-fill in the response slot is aligned with the multi-fill in the key slot that results in the best multi-fill-to-multi-fill F-score. Any leftover multi-fills in the key slot are unscored, and are tallied as "noncommittal".

Key objects are aligned with response objects of the same object "type" or "class" To choose which objects are paired, the scorer first generates all possible pairs of objects in the class. The F-score for each pair of objects is calculated from the way

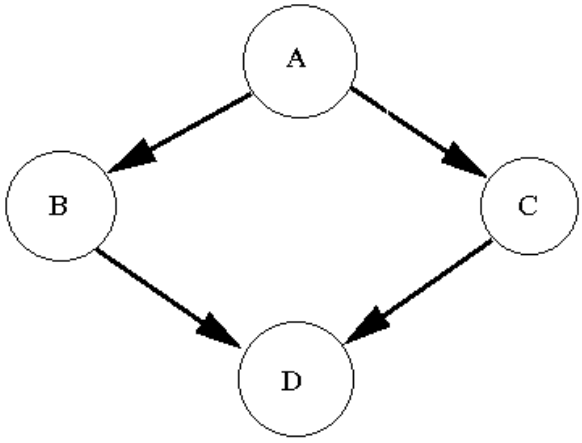
the objects' single-fills align. The weighted F-score is also calculated, by multiplying each slot-pair's F-score by the mapping weight of that slot, and summing the factors. The object pairs are sorted by (unweighted) F-score in decreasing order. Then the scorer proceeds down the sorted list, picking out pairs of objects for which neither single-fill has been chosen yet, and for which the weighted F-score exceeds the threshold for that type of object.

If any objects are left over after this, the scorer looks for any key objects which are marked "optional". The single fills of these objects are tallied as non-committal. If any key objects are left after this, their single-fills are tallied as missing. The single fills of any leftover response objects are tallied as spurious.

When all classes of objects have been aligned, the tallies are summed, and the resulting measures are calculated.

Template Relation (TR) and Scenario Template (ST) Scoring

For the TR and ST tasks, the scoring proceeds just as in TE scoring, but the order of alignment of objects is important. It is helpful to look at the classes of objects in a TR or ST task as vertices of a topological graph. If one type of object has a slot containing pointers to another type of object, then the graph has a directed edge from the first class to the pointed-to class:



When comparing a key pointer fill to a response pointer fill, the only way the scorer can compare the pointers is by looking to see if the objects to which they point have already been aligned by the scorer. If they have, and if the object pointed to by the key pointer is aligned to the object pointed to by the response pointer, then the pointers are tallied as correct.

Since pointer correctness is defined in this way, the directed graph cannot have any directed cycles in it. Further, the scorer has to align the objects so that any pointed-to objects must already have been aligned. So in the above figure, the order of mapping could be D-B-C-A or D-C-B-A. Any other order would confuse the scorer.

The only other difference between the TR and ST task and the TE task is the existence of *implicitly optional* objects in the key. In TR, a "relation" object that points to an optional "template element" object is optional, whether it's marked optional or not. And in ST, an object is implicitly optional if the only pointers pointing to that object are in optional slots or in one one multi-fill of a slot, but not in another multi-fill of the same slot (ie, there is an alternative multi-fill in the slot that doesn't point to the object).

Named Entity (NE) Task Scoring

The Named Entity task is scored like the Template Element task, except that the objects which are aligned must come from SGML elements in the same position of the original text file. For instance, if in the key the name "Bill Clinton" is tagged in the first paragraph of an article, and in the response "Bill Clinton" is tagged in the tenth paragraph, the objects will not be aligned, even if they would give an F-score of 100%.

Coreference (CO) Task Scoring

The scoring of the Coreference task is very different from that of the other four tasks. Rather than counting single fills, the CO algorithm compares equivalence classes of objects in the key with equivalence classes of objects in the the response. For a detailed explanation, see A Model-Theoretic Coreference Scoring Scheme, by Mark Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman in the MUC-6 Proceedings.
