A Practical Methodology for the Evaluation of Spoken Language Systems

Sean Boisen and Madeleine Bates*
Bolt Beranek and Newman, Inc. (BBN)
10 Moulton Street, Cambridge MA 02138 USA
sboisen@bbn.com, bates@bbn.com

1 Introduction

A meaningful evaluation methodology can advance the state-of-the-art by encouraging mature, practical applications rather than "toy" implementations. Evaluation is also crucial to assessing competing claims and identifying promising technical approaches. While work in speech recognition (SR) has a history of evaluation methodologies that permit comparison among various systems, until recently no methodology existed for either developers of natural language (NL) interfaces or researchers in speech understanding (SU) to evaluate and compare the systems they developed.

Recently considerable progress has been made by a number of groups involved in the DARPA Spoken Language Systems (SLS) program to agree on a methodology for comparative evaluation of SLS systems, and that methodology has been put into practice several times in comparative tests of several SLS systems. These evaluations are probably the only NL evaluations other than the series of Message Understanding Conferences (Sundheim, 1989; Sundheim, 1991) to have been developed and used by a group of researchers at different sites, although several excellent workshops have been held to study some of these problems (Palmer *et al.*, 1989; Neal *et al.*, 1991).

This paper describes a practical "black-box" methodology for automatic evaluation of question-answering NL systems. While each new application domain will require some development of special resources, the heart of the methodology is domain-independent, and it can be used with either speech or text input. The particular characteristics of the approach are described in the following section: subsequent sections present its implementation in the DARPA SLS community, and some problems and directions for future development.

2 The Evaluation Framework

2.1 Characteristics of the Methodology

The goal of this research has been to produce a well-defined, meaningful evaluation methodology which is

- automatic, to enable evaluation over large quantities of data
- based on an objective assessment of the understanding capabilities of a NL system (rather than its user interface, portability, speed, etc.)
- capable of application to a wide variety of NL systems and approaches
- suitable for blind testing
- as non-intrusive as possible on the system being evaluated (to decrease the costs of evaluation)
- domain independent.

The systems are assumed to be front ends to an interactive database query system, implemented in a particular common domain.

The methodology can be described as "black box" in that there is no attempt to evaluate the internal representations (syntactic, semantic, etc.) of a system. Instead, only the content of an answer retrieved from the database is evaluated: if the answer is correct, it is assumed that the system understood the query correctly. Comparing answers has the practical advantage of being a simple way to give widely varied systems a common basis for comparison. Although some recent work has suggested promising approaches (Black *et al.*, 1991), system-internal representations are hard to compare, or even impossible in some cases where System X has no level of representation corresponding to System Y's. It is easy, however, to define a simple common language for representing answers (see Appendix A), and easy to map system-specific representations into this common language.

This methodology has been successfully applied in the context of cross-site blind tests, where the evaluation is based on input which the system has never seen before. This type of evaluation leaves out many other important aspects of a system, such as the user interface, or the utility (or speed) of performing a particular task with a system that includes a NL component (work by Tennant (1981), Bates and Rettig (1988), and Neal *et al.* (1991) addresses some of these other factors).

Examples below will be taken from the current DARPA SLS application, the Airline Travel Information System (ATIS). This is a database of flights with information on the aircraft, stops and connections, meals, etc.

*The work reported here was supported by the Advanced Research Projects Agency and was monitored by the Office of Naval Research under Contract No. 00014–89–C–0008. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the United States Government.

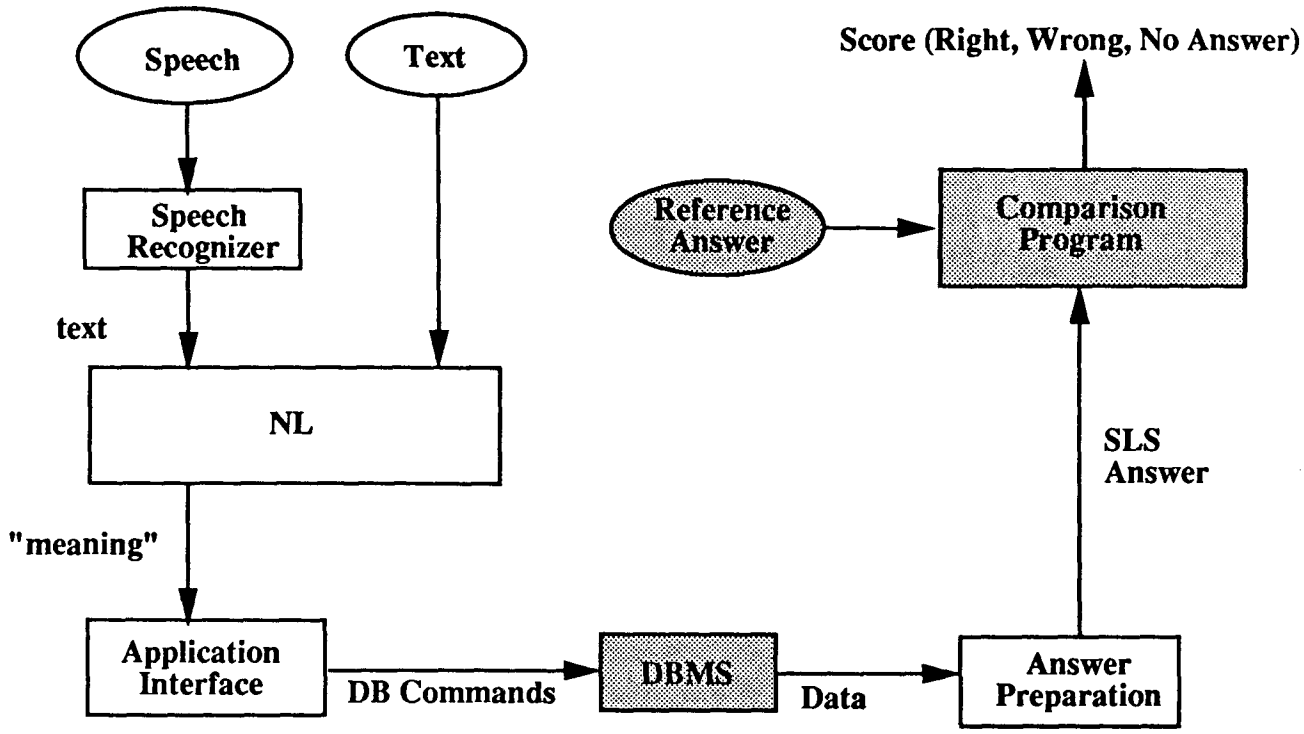

Figure 1: The evaluation process

2.2 Evaluation Architecture and Common Resources

We assume an evaluation architecture like that in Figure 1. The shaded components are common resources of the evaluation, and are not part of the system(s) being evaluated. Specifically, it is assumed there is a common database which all systems use in producing answers, which defines both the data tuples (rows in tables) and the data types for elements of these tuples (string, integer, etc.).

Queries relevant to the database are collected under conditions as realistic as possible (see 2.4). Answers to the corpus of queries must be provided, expressed in a common standard format (Common Answer Specification, or CAS): one such format is exemplified in Appendix A. Some portion of these pairs of queries and answers is then set aside as a test corpus, and the remainder is provided as training material.

In practice, it has also proved useful to include in the training data the database query expression (for example, an SQL expression) which was used to produce the reference answer: this often makes it possible for system developers to understand what was expected for a query, even if the answer is empty or otherwise limited in content.

2.2.1 Agreeing on Meaning

While the pairing of queries with answers provides the training and test corpora, these must be augmented by common agreement as to how queries should be answered. In practice, agreeing on the meaning of queries has been one of the hardest tasks. The issues are often extremely subtle, and interact with the structure and content of the database in sometimes unexpected ways.

As an example of the problem, consider the following request to an airline information system:

```
List the direct flights from
Boston to Dallas that serve
meals.
```

It seems straightforward, but should this include flights that might stop in Chicago without making a connection there? Should it include flights that serve a snack, since a snack is not considered by some people to be a full meal?

Without some common agreement, many systems would produce very different answers for the same questions, all of them equally right according to each system's own definitions of the terms, but not amenable to automatic inter-system comparison. To implement this methodology for such a domain, therefore, it is necessary to stipulate the meaning of potentially ambiguous terms such as "mid-day", "meals" , "the fare of a flight". The current list of such "principles of interpretation" for the ATIS domain contains about 60 specifications, including things like:

- which tables and fields in the database identify the major entities in the domain (flights, aircraft, fares, etc.)
- how to interpret fare expressions like "one-way fare", "the cheapest fare", "excursion fare", etc.
- which cities are to be considered "near" an airport.

Some other examples from the current principles of interpretation are given in Appendix B.

2.2.2 Reference Answers

It is not enough to agree on meaning of queries in the chosen domain. It is also necessary to develop a common understanding of precisely what is to be produced as the answer, or part of the answer, to a question.

For example, if a user asks "What is the departure time of the earliest flight from San Francisco to Atlanta?", one system might reply with a single time and another might reply with that time plus additional columns containing the carrier and flight number, a third system might also include the arrival time and the origin and destination airports. None of these answers could be said to be wrong, although one might argue about the advantages and disadvantages of terseness and verbosity.

While it is technically possible to mandate exactly which columns from the database should be returned for expressions, this is not practical: it requires agreement on a much larger set of issues, and conflicts with the principle that evaluation should be as non-intrusive as possible. Furthermore, it is not strictly necessary: what matters most is not whether a system provided exactly the same data as some reference answer, but whether the correct answer is clearly among the data provided (as long as no incorrect data was returned).

For the sake of automatic evaluation, then, a canonical reference answer (the minimum "right answer") is developed for each evaluable query in the training set. The content of this reference answer is determined both by domain-independent linguistic principles (Boisen *et al.*, 1989) and domain-specific stipulation. The language used to express the answers for the ATIS domain is presented in Appendix A.

Evaluation using the minimal answer alone makes it possible to exploit the fact that extra fields in an answer are not penalized. For example, the answer

```
(("AA" 152 0920 1015 "BOS" "CHI"
  "SNACK"))
```

could be produced for any of the following queries:

- "When does American Airlines flight 152 leave?"
- "What's the earliest flight from Boston to Chicago?"
- "Does the 9:20 flight to Chicago serve meals?"

and would be counted correct.

For the ATIS evaluations, it was necessary to rectify this problem without overly constraining what systems can produce as an answer. The solution arrived at was to have two kinds of reference answers for each query: a minimum answer, which contains the absolute minimum amount of data that must be included in an answer for it to be correct, and a maximum answer (that can be automatically derived from the minimum) containing all the "reasonable" fields that might be included, but no completely irrelevant ones. For example, for a question asking about the arrival time of a flight, the minimum answer would contain the flight ID and the arrival time. The maximum answer would contain the airline name and flight number, but not the meal service or any fare information. In order to be counted correct, the answer produced by a system must contain at least the data in the minimum answer, and no more than the data in the maximum answer; if additional fields are produced, the answer is counted as wrong. This successfully reduced the incentive for systems to overgenerate answers in hope of getting credit for answering queries that they did not really understand.

2.2.3 Comparison Software

Another common resource is software to compare the reference answers to those produced by various systems. [1] This task is complicated substantially by the fact that the reference answer is intentionally minimal, but the answer supplied by a system may contain extra information, and cannot be assumed to have the columns or rows in the same order as the reference answer. Some intelligence is therefore needed to determine when two answers match: simple identity tests won't work.

In the general case, comparing the atomic values in an answer expression just means an identity test. The only exception is real numbers, for which an epsilon test is performed, to deal with round-off discrepancies arising from different hardware precision.[2] The number of significant digits that are required to be the same is a parameter of the comparator.

Answer comparison at the level of tables require more sophistication, since column order is ignored, and the answer may include additional columns that are not in the specification. Furthermore, those additional columns can mean that the answer will include extra whole tuples not present in the specification. For example, in the ATIS domain, if the Concorde and Airbus are both aircraft whose type is "JET", they would together contribute only one tuple (row) to the simple list of aircraft types below.

```
(("JET")
 ("TURBOPROP")
 ("HELICOPTER")
 ("AMPHIBIAN")
 ("PROPELLER"))
```

On the other hand, if aircraft names were included in the table, they would each appear, producing a larger number of tuples overall.

```
(("AEROSPATIALE CONCORDE" "JET")
 ("AIRBUS INDUSTRIE" "JET")
 ("LOCKHEED L188 ELECTRA" "TURBOPROP")
 ...)
```

With answers in the form of tables, the algorithm explores each possible mapping from the required columns found in the reference answer (henceforth REF) to the actual columns found in the answer being evaluated (HYP). (Naturally, there must be at least as many columns in HYP as in REF, or the answer is clearly wrong.) For each such mapping, it reduces HYP according to the mapping, eliminating any duplicate tuples in the reduced table, and then compares REF against that reduced table, testing set-equivalence between the two.

Special provision is made for single element answers, so that a scalar REF and a HYP which is a table containing a single element are judged to be equivalent. That is, a scalar REF will match either a scalar or a single element

[1] The first implementation of this software was by Lance Ramshaw (Boisen *et al.*, 1989). It has since been re-implemented and modified by NIST for the ATIS evaluations.

[2] For the ATIS evaluations, this identity test has been relaxed somewhat so that, e.g., strings need not have quotes around them if they do not contain "white space" characters. See Appendix A for further details.

table for HYP, and a REF which is a single element table specification will also match either kind of answer.

For the ATIS evaluations, two extensions were made to this approach. A REF may be ambiguous, containing several sub expressions each of which is itself a REF: in this case, if HYP matches any of the answers in REF, the comparison succeeds. A special answer token (NO_ANSWER) was also agreed to, so that when a system can detect that it doesn't have enough information, it can report that fact rather than guessing. This is based on the assumption that failing to answer is less serious than answering incorrectly.

2.3 Scoring Answers

Expressing results can be almost as complicated as obtaining them. Originally it was thought that a simple "X percent correct" measure would be sufficient, however it became clear that there was a significant difference between giving a wrong answer and giving no answer at all, so the results are now presented as: Number right, Number wrong, Number not answered, Weighted Error Percentage (weighted so that wrong answers are twice as bad as no answer at all), and Score (100 - weighted error).

Whenever numeric measures of understanding are presented, they should in principle be accompanied by some measure of the significance and reliability of the metric. Although precise significance tests for this methodology are not yet known, it is clear that "black box" testing is not a perfect measure. In particular, it is impossible to tell whether a system got a correct answer for the "right" reason, rather than through chance: this is especially true when the space of possible answers is small (yes-no questions are an extreme answer). Since more precise measures are much more costly, however, the present methodology has been considered adequate for the current state of the art in NL evaluation.

Given that current weighted error rates for the DARPA ATIS evaluations range from 55%–18%, we can roughly estimate the confidence interval to be approximately 8%.[3]

Another source of variation in the scoring metric is the fact that queries taken from different speakers can vary widely in terms of how easy it is for systems to understand and answer them correctly. For example, in the February 1991 ATIS evaluations, the performance of BBN's Delphi SLS on text input from individual speakers ranged from 75% to 10% correct. The word error from speech recognition was also the highest for those speakers with the highest NL error rates, suggesting that individual speaker differences can strongly impact the results.

[3]Assuming there is some probability of error in each trial (query), the variance in this error rate can be estimated using the formula

$$2\sqrt{\frac{e(1-e)}{n}}$$

where e is the error rate expressed as a decimal (so 55% error = 0.55), and n is the size of the test set. Taking $e = 0.45$ (one of the better scores from the February 91 ATIS evaluation), and $n = 145$, differences in scores greater than 0.08 (8%) have a 95% likelihood of being significant.

2.4 Evaluation Data

2.4.1 Collecting Data

The methodology presented above places no *a priori* restrictions on how the data itself should be collected. For the ATIS evaluations, several different methods of data collection, including a method called "Wizard scenarios", were used to collect raw data, both speech and transcribed text (Hemphill, 1990). This resulted in the collection of a number of human-machine dialogues. One advantage of this approach is that it produced both the queries and draft answers at the same time. It also became clear that the language obtained is very strongly influenced by the particular task, the domain and database being used, the amount and form of data returned to the user, and the type of data collection methodology used. This is still an area of active research in the DARPA SLS community.

2.4.2 Classifying Data

Typically, some of the data which is collected is not suitable as test data, because:

- the queries fall outside the domain or the database query application
- the queries require capabilities beyond strict NL understanding (for example, very complex inferencing or the use of large amounts of knowledge outside the domain)
- the queries are overly vague ("Tell me about ...")

It is also possible that phenomena may arise in test data which falls outside the agreement on meanings derived from the training data (the "principles of interpretation"). Such queries should be excluded from the test corpus, since it is not possible to make a meaningful comparison on answers unless there is prior agreement on precisely what the answer should be.

2.4.3 Discourse Context

The methodology of comparing paired queries and answers assumes the query itself contains all the information necessary for producing an answer. This is, of course, often not true in spontaneous goal-directed utterances, since one query may create a context for another, and the full context is required to answer (e.g., "Show me the flights ...", "Which of THEM ..."). Various means of extending this methodology for evaluating context-dependent queries have been proposed, and some of them have been implemented in the ATIS evaluations (Boisen *et al.* (1989), Hirschman *et al.* (1990), Bates and Ayuso (1991), Pallett (1991)).

3 The DARPA SLS Evaluations

The goal of the DARPA Spoken Language Systems program is to further research and demonstrate the potential utility of speech understanding. Currently, at least five major sites (AT&T, BBN, CMU, MIT, and SRI) are developing complete SLS systems, and another site (Paramax) is integrating its NL component with other speech systems. Representatives from these and other organizations meet regularly to discuss program goals and to evaluate progress.

This DARPA SLS community formed a committee on evaluation[4], chaired by David Pallett of the National Institute of Standards and Technology (NIST). The committee was to develop a methodology for data collection, training data dissemination, and testing for SLS systems under development. The first community-wide evaluation using the first version of this methodology took place in June, 1990, with subsequent evaluations in February 1991 and February 1992.

The emphasis of the committee's work has been on automatic evaluation of queries to an air travel information system (ATIS). Air travel was chosen as an application that is easy for everyone to understand. The methodology presented here was originally developed in the context of the need for SLS evaluation, and has been extended in important ways by the community based on the practical experience of doing evaluations.

As a result of the ATIS evaluations, a body of resources has now been compiled and is available through NIST. This includes the ATIS relational database, a corpus of paired queries and answers, protocols for data collection, software for automatic comparison of answers, the "Principles of Interpretation" specifying domain-specific meanings of queries, and the CAS format (Appendix A is the current version). Interested parties should contact David Pallet of NIST for more information.[5]

4 Advantages and Limitations of the Methodology

Several benefits come from the use of this methodology:

- It forces advance agreement on the meaning of critical terms and on some information to be included in the answer.
- It is objective, to the extent that a method for selecting testable queries can be defined, and to the extent that the agreements mentioned above can be reached.
- It requires less human effort (primarily in the creating of canonical examples and answers) than non-automatic, more subjective evaluation. It is thus better suited to large test sets.
- It can be easily extended.

Most of the weaknesses of this methodology arise from the fact that the answers produced by a database query system are only an approximation of its understanding capabilities. As with any black-box approach, it may give undue credit to a system that gets the right answer for the wrong reason (i.e., without really understanding the query), although this should be mitigated by using larger and more varied test corpora. It does not distinguish between merely acceptable answers and very good answers.

Another limitation of this approach is that it does not adequately measure the handling of some phenomena, such as extended dialogues.

5 Other Evaluation Methodologies

This approach to evaluation shares many characteristics with the methods used for the DARPA-sponsored Message Understanding Conferences (Sundheim, 1989; Sundheim, 1991). In particular, both approaches are focused on external (black-box) evaluation of the understanding capabilities of systems using input/output pairs, and there are many similar problems in precisely specifying how NL systems are to satisfy the application task.

Despite these similarities, this methodology probably comes closer to evaluating the actual understanding capabilities of NL systems. One reason is that the constraints on both input and output are more rigorous. For database query tasks, virtually every word must be correctly understood to produce a correct answer: by contrast, much of the MUC-3 texts is irrelevant to the application task. Since this methodology focuses on single queries (perhaps with additional context), a smaller amount of language is being examined in each individual comparison.

Similarly, for database query, the database itself implicitly constrains the space of possible answers, and each answer is scored as either correct or incorrect. This differs from the MUC evaluations, where an answer template is a composite of many bits of information, and is scored along the dimensions of recall, precision, and overgeneration.

Rome Laboratory has also sponsored a recent effort to define another approach to evaluating NL systems (Neal *et al.*, 1991; Walter, 1992). This methodology is focussed on human evaluation of interactive systems, and is a "glass-box" method which looks at the performance of the linguistic components of the system under review.

6 Future Issues

The hottest topic currently facing the SLS community with respect to evaluation is what to do about dialogues. Many of the natural tasks one might do with a database interface involve extended problem-solving dialogues, but no methodology exists for evaluating the capabilities of systems attempting to engage in dialogues with users.

A Common Answer Specification (CAS) for the ATIS Application

(Note: this is the official CAS specification for the DARPA ATIS evaluations, as distributed by NIST. It is domain independent, but not necessarily complete: for example, it assumes that the units of any database value are unambiguously determined by the database specification. This would not be sufficient for applications that allowed unit conversion, e.g. "Show me the weight of ..." where the weight could be expressed in tons, metric tons, pounds, etc. This sort of extension should not affect the ease of automatically comparing answer expressions, however.)

[4]The primary members of the original committee are: Lyn Bates (BBN), Debbie Dahl (UNISYS), Bill Fisher (NIST), Lynette Hirschman (MIT), Bob Moore (SRI), and Rich Stern (CMU). Successor committees have also included Jared Bernstein (SRI), Kate Hunike-Smith (SRI), Patti Price (SRI), Alex Rudnicky (CMU), and Jay Wilpon (AT&T). Many other people have contributed to the work of these committees and their subcommittees.

[5]David Pallet may be contacted at the National Institute of Standards and Technology, Technology Building, Room A216, Gaithersburg, MD 20899, (301)975–2944.

Basic Syntax in BNF

answer ⟶ *cas1* | (*cas1* **OR** *answer*)

cas1 ⟶ *scalar-value* | *relation* | **NO_ANSWER** | **no_answer**

scalar-value ⟶ *boolean-value* | *number-value* | *string*

boolean-value ⟶ **YES** | **yes** | **TRUE** | **true** | **NO** | **no** | **FALSE** | **false**

number-value ⟶ *integer* | *real-number*

integer ⟶ *[sign] digit+*

sign ⟶ **+** | **-**

digit ⟶ **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9**

real-number ⟶ *sign digit+ . digit** | *digit+ . digit**

string ⟶ *char_except_whitespace+* | " *char** "

relation ⟶ (*tuple**)

tuple ⟶ (*value+*)

value ⟶ *scalar-value* | **NIL**

Standard BNF notation has been extended to include two other common devices : "A+" means "one or more A's" and "A*" means "zero or more A's".

The formulation given above does not define *char_except_whitespace* and *char*. All of the standard ASCII characters count as members of *char*, and all but "white space" are counted as *char_except_whitespace*. Following ANSI "C", blanks, horizontal and vertical tabs, newlines, formfeeds, and comments are, collectively, "white space".

The only change in the syntax of CAS itself from the previous version is that now a string may be represented as either a sequence of characters not containing white space or as a sequence of any characters enclosed in quotation marks. Note that only non-exponential real numbers are allowed, and that empty tuples are not allowed (but empty relations are).

Additional Syntactic Constraints

The syntactic classes *boolean-value*, *string*, and *number-value* define the types "boolean", "string", and "number", respectively. All the tuples in a relation must have the same number of values, and those values must be of the same respective types (boolean, string, or number).

If a token could represent either a string or a number, it will be taken to be a number; if it could represent either a string or a boolean, it will be taken to be a boolean. Interpretation as a string may be forced by enclosing a token in quotation marks.

In a tuple, **NIL** as the representation of missing data is allowed as a special case for any value, so a legal answer indicating the costs of ground transportation in Boston would be

```
(("L" 5.00) ("R" nil)
 ("A" nil) ("R" nil))
```

Elementary Rules for CAS Comparisons

String comparison is case-sensitive, but the distinguished values (**YES**, **NO**, **TRUE**, **FALSE**, **NO_ANSWER**, and **NIL**) may be written in either upper or lower case.

Each indexical position for a value in a tuple (say, the *i*th) is assumed to represent the same field or variable in all the tuples in a given relation.

Answer relations must be derived from the existing relations in the database, either by subsetting and combining relations or by operations like averaging, summation, etc.

In matching an hypothesized (HYP) CAS form with a reference (REF) one, the order of values in the tuples is not important; nor is the order of tuples in a relation, nor the order of alternatives in a CAS form using **OR**. The scoring algorithm will use the re-ordering that maximizes the indicated score. Extra values in a tuple are not counted as errors, but distinct extra tuples in a relation are. A tuple is not distinct if its values for the fields specified by the REF CAS are the same as another tuple in the relation; these duplicate tuples are ignored. CAS forms that include alternate CAS's connected with **OR** are intended to allow a single HYP form to match any one of several REF CAS forms. If the HYP CAS form contains alternates, the score is undefined.

In comparing two real number values, a tolerance will be allowed; the default is ±.01%. No tolerance is allowed in the comparison of integers. In comparing two strings, initial and final sub-strings of white space are ignored. In comparing boolean values, **TRUE** and **YES** are equivalent, as are **FALSE** and **NO**.

B Some Examples from the Principles of Interpretation Document for the ATIS Application

(Note: these are excerpted from the official Principles of Interpretation document dated 11/20/91. The entire document is comprised of about 60 different points, and is available from David Pallet at NIST.

The term "annotator" below refers to a human preparing training or test data by reviewing reference answers to queries.)

INTERPETING ATIS QUERIES RE THE DATABASE

1 General Principles:

1.1 Only reasonable interpretations will be used.

An annotator or judge must decide if a linguistically possible interpretation is reasonable or not.

1.2 The context will be used in deciding if an interpretation is reasonable.

...

1.3 Each interpretation must be expressible as one SQL statement.

At present (11/18/91) a few specified exceptions to this principle are allowed, such as allowing boolean answers for yes/no questions.

1.4 All interpretations meeting the above rules will be used by the annotators to generate possible reference answers.

A query is thus ambiguous iff it has two interpretations that are fairly represented by distinct SQL expressions.

The reference SQL expression stands as a semantic representation or logical form. If a query has two interpretations that result in the same SQL, it will not be considered ambiguous. The fact that the two distinct SQL expressions may yield the same answer given the database is immaterial.

The annotators must be aware of the usual sources of ambiguity, such as structural ambiguity, exemplified by cases like "the prices of flights, first class, from X to Y", in which the attachment of a modifier that can apply to either prices or flights is unclear. (This should be (ambiguously) interpreted both ways, as both "the first-class prices on flights from X to Y" and "the prices on first-class flights from X to Y".) More generally, if structural ambiguities like this could result in different (SQL) interpretations, they must be treated as ambiguous.

...

2 Specific Principles:

In this arena, certain English expressions have special meanings, particularly in terms of the database distributed by TI in the spring of 1990 and revised in November 1990 and May 1991. Here are the ones we have agreed on: (In the following, "A.B" refers to field B of table A.)

2.1 Requests for enumeration.

A large class of tables in the database have entries that can be taken as defining things that can be asked for in a query. In the answer, each of these things will be identified by giving a value of the primary key of its table. These tables are:

Table Name	English Term(s)	Primary Key
aircraft	aircraft, equipment	aircraft_code
airline	airline	airline_code
airport	airport	airport_code
...		
flight_stop	(intermed.) stops	flight_id, stop_number high_flight_number

2.2 Flights.

2.2.1 A flight "between X and Y" means a flight "from X to Y".

...

2.2.3 A request for a flight's stops will be interpreted as asking for the intermediate stops only, from the flight_stop table.

...

2.3 Fares.

2.3.1 A "one-way" fare is a fare for which round_trip_required = "NO".

2.3.2 A "round-trip" fare is a fare with a non-null value for fare.round_trip_cost.

2.3.3 The "cheapest fare" means the lowest one-direction fare.

...

2.3.8 Questions about fares will always be treated as fares for flights in the maximal answer.

2.4 Times.

2.4.1 The normal answer to otherwise unmodified "when" queries will be a time of day, not a date or a duration.

2.4.2 The answer to queries like "On what days does flight X fly" will be a list of days.day_name fields.

2.4.3 Queries that refer to a time earlier than 1300 hours without specifying "a.m." or "p.m." are ambiguous and may be interpreted as either.

2.4.4 Periods of the day.

The following table gives precise interpretations for some vague terms referring to time periods. The time intervals given do not include the end points. Items flagged with "*" are in the current (rdb3.3) database interval table.

PERIOD	BEGIN TIME	END TIME
morning*	0000	1200
afternoon*	1200	1800
evening*	1800	2200
day*	600	1800
night*	1800	600
early morning*	0000	800

...

...

2.9 Meaning requests.

2.9.1 With the particular exceptions noted below, requests for the "meaning" of something will only be interpretable if that thing is a code with a canned decoding definition in the database. In case the code field is not the key field of the table, information should be returned for all tuples that match on the code field. Here are the things so defined, with the fields containing their decoding:

Table	Code Field	Decoding Field
aircraft	aircraft_code	aircraft_description
airline	airline_code	airline_name
airport	airport_code	airport_name
city	city_code	city_name
class_of_service	booking_class	class_description
code_description	code	description

...

2.11 Queries that are literally yes-or-no questions are considered to be ambiguous between interpretation as a yes-or-no question and interpretation as the corresponding wh-question. For example, "Are there flights from Boston to Philly?" may be answered by either a boolean value ("YES/TRUE/NO/FALSE") or a table of flights from Boston to Philadelphia.

2.15 When a query refers to an aircraft type such as "BOEING 767", the manufacturer (if one is given) must match the aircraft.manufacturer field and the type may be matched against either the aircraft.code field or the aircraft.basic_type field, ambiguously.

2.16 Utterances whose answers require arithmetic computation are not now considered to be interpretable; this does not apply to arithmetic comparisons, including computing the maximum or minimum value of a field, or counting elements of a set of tuples.

...

References

B. Ballard. *A Methodology for Evaluating Near-Prototype NL Processors.* Technical Report OSU–CISRC–TR-81-4, Ohio State University, 1981.

M. Bates and D. Ayuso. A proposal for incremental dialogue evaluation. In *Proceedings of the Speech and Natural Language Workshop*, San Mateo, California, February 1991. DARPA, Morgan Kaufmann Publishers, Inc.

M. Bates and M. Rettig. How to choose NL software. *AI Expert*, July 1988.

E. Black, S. Abney, D. Flickenger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, B. Ingria, F. Jelinek, J. Klavens, M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proceedings of the Speech and Natural Language Workshop*, San Mateo, California, February 1991. DARPA, Morgan Kaufmann Publishers, Inc.

S. Boisen, L. Ramshaw, D. Ayuso, and M. Bates. A proposal for SLS evaluation. In *Proceedings of the Speech and Natural Language Workshop*, San Mateo, California, October 1989. DARPA, Morgan Kaufmann Publishers, Inc.

DARPA. *Proceedings of the Speech and Natural Language Workshop*, San Mateo, California, June 1990. Morgan Kaufmann Publishers, Inc.

DARPA. *Proceedings of the Speech and Natural Language Workshop*, San Mateo, California, February 1991. Morgan Kaufmann Publishers, Inc.

DARPA. *Proceedings of the Third Message Understanding Conference (MUC-3)*, San Mateo, California, May 1991. Morgan Kaufmann Publishers, Inc.

DARPA. *Proceedings of the Speech and Natural Language Workshop*, San Mateo, California, February 1992. Morgan Kaufmann Publishers, Inc.

C. Hemphill. TI implementation of corpus collection. In *Proceedings of the Speech and Natural Language Workshop*, San Mateo, California, June 1990. DARPA, Morgan Kaufmann Publishers, Inc.

L. Hirschman, D. Dahl, D. McKay, L. Norton, and M. Linebarger. A proposal for automatic evaluation of discourse. In *Proceedings of the Speech and Natural Language Workshop*, San Mateo, California, June 1990. DARPA, Morgan Kaufmann Publishers, Inc.

J. Neal, T. Finin, R. Grishman, C. Montgomery, and S. Walter. *Workshop on the Evaluation of Natural Language Processing Systems.* Technical Report (to appear), RADC, June 1991.

D. S. Pallett. DARPA Resource Management and ATIS benchmark test poster session. In *Proceedings of the Speech and Natural Language Workshop*, San Mateo, California, February 1991. DARPA, Morgan Kaufmann Publishers, Inc.

M. Palmer, T. Finin, and S. Walter. *Workshop on the Evaluation of Natural Language Processing Systems.* Technical Report RADC-TR-89-302, RADC, 1989.

B. M. Sundheim. Plans for a task-oriented evaluation of natural language understanding systems. In *Proceedings of the Speech and Natural Language Workshop*, pages 197–202, Philadelphia, PA, Februrary 1989.

B. M. Sundheim. Overview of the Third Message Understanding Evaluation and Conference. In *Proceedings of the Third Message Understanding Conference (MUC-3)*, pages 3–16, San Mateo, California, May 1991. DARPA, Morgan Kaufmann Publishers, Inc.

H. Tennant. *Evaluation of Natural Language Processors.* PhD thesis, University of Illinois, 1981.

S. Walter. Neal-Montgomery NLP system evaluation methodology. In *Proceedings of the Speech and Natural Language Workshop*, San Mateo, California, February 1992. DARPA, Morgan Kaufmann Publishers, Inc.

R. M. Weischedel. *Issues and Red Herrings in Evaluating Natural Language Interfaces.* Pergamnon Press, 1986.