The Vocabulary Problem in Human-System Communication: an Analysis and a Solution

G. W. Furnas
T. K. Landauer
L. M. Gomez
S. T. Dumais

Bell Communications Research
E

white.me.out[1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14]

1. Introduction

Many functions of most large systems depend on users typing in the right words. New or intermittent users often use the wrong words and fail to get the actions or information they want. This is the vocabulary problem. It is a troublesome impediment in computer interactions both simple (file access and command entry) and complex (database query and natural language dialog).

In what follows we report evidence on the extent of the vocabulary problem, and propose both a diagnosis and a cure. The fundamental observation is that people use a surprisingly great variety of words to refer to the same thing. In fact, the data show that no single access word, however well chosen, can be expected to cover more than a small proportion of user's attempts. Designers have almost always underestimated the problem and, by assigning far too few alternate entries to databases or services, created an unnecessary barrier to effective use. Simulations and direct experimental tests of several alternative solutions show that rich, probabilistically weighted indexes or alias lists can improve success rates by factors of three to five.

2. The vocabulary problem as variability in word usage

If everyone always agreed on what to call things, the user's word would be the designer's word would be the system's word, and what the user typed or pointed to would be mutually understood. Unfortunately, people often disagree on the words they use for things. To experience the problem, try the exercise in Figure 1. It is given here in the context of a command access ("command naming") problem, but it could just as well have come from library information retrieval or database query.

Insert Figure 1 About Here

Ask one other colleague to try it too and compare your responses.

We have asked this of more than a thousand pairs of people, including programmers, computer science students, and human factors specialists. Less than a dozen pairs have agreed. This means that without tutoring, less than a dozen people out of a thousand could have directly accessed the command, had their partner named it. (As a follow-up exercise, list other good names you think of. You will be able to compare your list against one we reveal later.)

In current computer systems, the vocabulary problem is largely ignored. Designers decide on the terms to be used, and as heavy users, grow to find these terms obvious and natural. Other users are simply required to learn the system's words. [1]

In information retrieval systems, the keywords that are assigned by indexers are often at odds with those tried by searchers. The seriousness of the problem is indicated by the need for professional intermediaries between users and systems, and by disappointingly low average performance (recall) rates.

The standard conceptualization of the naming problem is to begin with the system's objects or functions and ask what word or words should be associated with each: What name for this command? What keyword for this document? We will argue later (section 3.5) that this approach is basically misleading, but we will explore it first. Throughout we take an empirical approach, collecting large amounts of data on actual human language usage, then modeling and evaluating different system strategies. A more detailed exposition of the data collection and analysis methods can be found in [4].

[1] Landauer, Gallotti and Hartwell[8] found that the initial learning of an editor by beginners was not much effected by the differences in the "naturalness" of the small number of terms they needed to learn. However, learning a few names is not the major problem in mastering one's first computer application, while knowing the right terms may be a very large factor in the effective use of extensive systems.

3. The Data

In an attempt to identify some basic principles of broad generality, we collected data on five very different but typical application domains. The words in square brackets are titles for the data sets, and will be used in the tables that follow.

(1) Main verbs applied by 48 typists to describe manual text editing operations, for 5 generic operations or 25 combinations of operations and text units [Editor-5, Editor-25]. The typists saw author-marked corrections and provided brief instructions as if to another typist.

(2) Commands for a hypothetical "message decoder" program ["Decoder"], nominated by 100 experienced system designers after studying the design and interface[2].

(3) The first content word used in describing a selection of 50 common objects ["Common objects"], given by 337 college students. They were shown short verbal descriptions such as "love", "motorcycle", or "Newsweek", and asked to give alternative words or phrases for a person or computer to understand.

(4) First-nominated superordinate category for 64 "Swap ‘n Sale" classified ad items ["Classifieds"], given by 30 New Jersey homemakers.

(5) First priority keyword nominated for a computerized file of 188 recipes ["Recipe Keywords"], by 8 expert cooks and 16 homemakers.

These domains all involved information objects that one might want to access on a computer and were all of modest size (5-200 objects), but in most other respects they differed: in knowledge domain, in elicitation technique, in type of people, in preprocessing (e.g., ignoring endings or non-content words). Note also that the first two relate to command access, the third to general knowledge description, and the last two to information retrieval. Conclusions that apply to all of these disparate domains must be based on general rather than idiosyncratic properties, and are therefore likely to hold for many others.

[2] Data from P. Barnard, Applied Psychology Unit, Cambridge, England. Personal communication.

In what follows we will use "object" or "item" to refer to any of the entities (text editor operations, decoder commands, common objects, advertised items, and cooking recipes) which we presented to people as stimuli. The words we got back we will call "terms", "words", "descriptors", or "names."

3.1 Analyses

For each domain, we counted how often each word was applied to each object. Samples of the resulting data are shown in Tables 1a and 1b. (Objects are denoted at the top of each column by abbreviated examples or descriptions; the actual 'objects' seen by the subjects, as summarized above, are given in [4].)

Insert Table 1 About Here

For example Table 1a indicates that 22 subjects proposed the word "change" as a descriptor for the editing operation indicated by a crossed out word by an author, and referred to in the table as *"delete"*.

All the tables were very sparse. One reason is that word usage tended to follow Zipf's distribution[14] -- a few words are used very frequently, the vast majority only rarely; more importantly however, most words are applied to only a few objects.

Because they estimate the likelihood of users or designers giving a word for an object, these tables allow us to simulate the performance of various approaches to word-based access. To see how untutored users' words would fare against a system, we can simply sample from these tables instead of going back to subjects.

3.2 ''Armchair'' naming

The simplest version of vocabulary-based access is when objects are accessible through a single special term - its "name". The most common strategy is for designers to install their own personal favorites as object names. We call this the "Armchair" design method. It gives users access only if their word coincides with the designer's.

For our data sets the probabilities that two people (e.g., a user and designer) coincide on the word they spontaneously apply to a given object is presented in Results Table 1.[3]

Results Table 1
Probability two people apply the same term to an object

Editor-5	Editor-25	Decoder	Common Objects	Classifieds	Recipe Keywords
.07	.11	.08	.12	.14	.18

Clearly people disagree greatly in the labels they use. The probability that two typists will use the same main verb in describing an editing operation is less than one in fourteen; that two cooks will use the same first keyword for a recipe is less than one in five.

The data tell us that the armchair method is highly unsatisfactory. If one person assigns the name of an item, other untutored people will *fail* to access it on 80% to 90% of their attempts. This finding is not only true of all six of our laboratory data sets; it has also been confirmed several times by research with actual systems (Furnas[4]; Gomez and Lochbaum [5]; Good, Whiteside, Wixon, and Jones[6]) .

One might be tempted to think that domain experts could do a better naming job. In the recipe study, one third of keyword providers were expert cooks. We found, however, that their keywords fared no better than average, either in indexing for other experts or for novices. Similarly the decoder command names were chosen by and for experts and had very low agreement.

So far we have assumed no requirement that access terms be distinct, that is, that each name be assigned to only one object. While in information retrieval contexts, several documents may share a keyword, for command or file names uniqueness is typically important: the system must have an unambiguous interpretation for the intended action and objects. Adding such a constraint obviously means that good candidate names for one object often get ruled out, having already been assigned to another object. As a result, untutored people will be even less likely to hit upon the official name.

In our data, requiring uniqueness caused proportional decrements of 5% to 60% (typically about 10%)

[3] We use an unbiased statistical estimator of this quantity, the *repeat rate*, given in [7].

in the already low performances. Constraining names to be multiple word strings, systematic, "cute," or even "mnemonic", is certain to result in even less likely first-try success. For example Good, et al.[6] found that their system's initial, armchair-named commands, were slightly below our predictions, presumably reflecting other constraints imposed by the system design on the name choices.

3.3 How good is the best possible name?

From the standpoint of first-try success for the untrained, the best possible access term would be the word real users most often apply to an object. This empirically based naming approach has been a popular proposal in human factors circles: terms offered by a representative sample of potential users would be collected, and the most frequent term identified.

We can identify this "best" name from data tables like Table 1. For statistical reasons[4], data can only provide estimates of bounds on the success rate of this method. A low estimate can be obtained by using a random half of the data to pick the best words, and the remaining half to test their effectiveness. A high estimate can be obtained by using the actual relative frequency in the data of the most popular word.

Results Table 2 presents the range of these low and high estimate hit rates for the data sets.

Results Table 2
Probability someone uses the most popular term for an object

Editor-5	Editor-25	Decoder	Common Objects	Classifieds	Recipe Keywords	
.15	.19	.22	.26	.26	.31	(low estimate)
.16	.22	.21	.28	.34	.36	(high estimate)

While there is typically about a 2 to 1 improvement over the straight armchair method, failures still occur 65-85% of the time.

Since no term will be hit upon more often than the most popular term, these numbers represent a true performance limit when a single access term is assigned to each object. Simply stated, the data tell us *there*

[4] There is no way to get an unbiased, distribution-free estimate of the "true" value of the maximum-frequency cell.

is no one good access term for most objects. The idea of an "obvious", "self-evident" or "natural" term is a myth! Since even the best possible name is not very useful, it follows that *there can exist no rules, guidelines or procedures for choosing a good name, in the sense of "accessible to the unfamiliar user".*

Any further improvement will require a different strategy.

3.4 The value of a few aliases

Once the idea of a single access term is rejected, the next logical suggestion is to try several access terms. Library catalogues and indices, for example, usually have at least two (but seldom more than four) entry points for each referenced object.

Below we present the estimated performance of a system where each object had three access terms known to the system, selected by frequency-weighted random sampling, mimicking the frequencies of a design group’s first three "armchair nominations."

Results Table 3
Probability someone uses one of three armchair aliases for an object

Editor-5	Editor-25	Decoder	Common Objects	Classifieds	Recipe Keywords
.21	.30	.20	.28	.34	.45

While the improvement over a single armchair name is considerable, three armchair aliases are no better than a single optimally chosen term.

If the three most popular, i.e., three optimally chosen access terms are used, the following results are obtained.

Results Table 4
Probability someone uses one of the three most popular aliases for an object

Editor-5	Editor-25	Decoder	Common Objects	Classifieds	Recipe Keywords	
.37	.42	.38	.42	.45	.58	(low estimate)
.38	.49	.41	.48	.59	.67	(high estimate)

This approach looks promising. Unfortunately, the marginal returns for even more aliases diminishes rapidly. Figure 2 presents an operating characteristic, plotting how many optimally selected aliases are needed to account for a given percentage of the untutored users’ attempts. The figure shows the results for the Common Objects data, which is typical of our data sets.

Insert Figure 2 About Here

The two curves present the high and low statistical estimates respectively. (Especially for larger numbers of aliases, practical limits are probably closer to the lower than the upper estimate.) Even with fifteen aliases, only 60-80% of "attempts" will be satisfied.

Clearly the only hope for untutored vocabulary driven access is to provide many, many alternate entry terms. Thus aliases are, indeed, the answer, but only if used on a much larger scale than usually considered.

The tendency to underestimate the need for so many aliases results, we conjecture, from the fact that any one person (for example, the designer) usually can think of only a handful of the terms that other people would consider appropriate. For the example exercise in Figure 1 at the beginning of the paper it is rare for one person to come up with more than a half dozen names. Across all people, however, over a hundred command names for that service have been suggested, including: activities, calendar, events, cityevents, whatsup, sparetime, funtime, weekender, nightout, downtown, entertain, outings, diversions, play, fun&games, amusements, guide, goingson, happenings, thingstodo, aroundtown, leisure, weekend, nightlife, abouttown, onthetown,....

3.5 A solution is unlimited aliasing

Our discussion of the "unfamiliar access" problem has followed the spirit of common practice and been dominated by a computer-centered point of view. Beginning with the set of system entities, various schemes for assigning access terms were proposed and evaluated. We believe this approach is misleading because it tends to focus on what the computer brings to the interaction (the system’s objects), yet it is a characteristic of what the human brings, the variability in vocabulary usage, that dominates the problem.

If we want the system to have the best chance of "understanding" the user, it is better to conceive of design the other way around; begin with user’s words and find system interpretations. We need to know, for every word that users will try in a given task environment, the relative frequencies with which they would be satisfied by various objects. The system would be designed using a table of word usage much like the ones we have collected. When users attempt access with a word, the highest frequency object for that word is the system’s best guess as to the user’s intent. For example in Table 1(b), when confronted by a user’s term "fruit", the best guess is a PEAR. If we had complete data this approach would yield the theoretically optimum performance for getting users and objects together.

In terms of the designer’s usual system-oriented standpoint, the proposal here is to allow essentially unlimited numbers of aliases. This is *not* to say that designers should build systems with unruly thousands of commands or objects in them. The number of distinct objects or commands needed is a matter of system functionality and good design. [5] But, regardless of the number of commands or objects in a system and whatever the choice of their "official" names, *the designer must make available many, many alternate verbal access routes to each.*

3.5.1 The precision problem

Before we discuss the practicality and estimated performance of this approach, we must face what is called, in information retrieval literature, the "precision" problem.

Whenever a given word is applied to more than one object, ambiguity results. For example in the data of Table 1b, PEAR (frequency 19) is the best guess for "fruit", but the user might also mean NECTARINE (frequency 12) or even RAISIN (frequency 1). The "fruit" example illustrates imprecision arising from the generality of a term. It may also arise from polysemy -- the same word may mean two or more distinct things. In a retrieval based on the term "nut" a system can only guess whether to return items related to

[5] In fact, careful system design may itself help the untutored’s vocabulary access problem. For example if a system’s intrinsic structure can be designed to support it, a factorial task decomposition can lead to predictable patterns of official terminology, perhaps allowing users to predict some terminology from others ("delete word", "transpose word", "delete sentence", ==> "transpose sentence").

hardware, seeds or eccentrics.

How serious is the precision problem? In our data the probabilities that two people intended the same referent(s) by a given term were:

Results Table 5
Probability two people using a term intend the same object

Editor-5	Editor-25	Decoder	Common Objects	Classifieds	Recipe Keywords
.41	.15	.73	.52	.62	.13

Here there was considerable variation between data sets. In the recipe data, two uses of a given keyword refer to the same recipe only about an eighth of the time, reflecting the fact that cooking terms often refer to several recipes. In contrast, two occurrences of a particular name for the hypothetical decoder referred to the same action three quarters of the time, indicating more restricted and precise usage.

There was a modest but reliable tendency within each data set for less frequent terms to be more precise[12]. Often more popular words, while being more likely for a given object, are also more likely for other objects (e.g., "change" in Table 1(a)). This observation is important since it means unlimited aliasing will not incur a great precision cost. Having more access words per object does not logically imply more objects per word, and empirically the average tendency appears to be the reverse. Moreover, in an interactive situation in which users can refine their selections by a series of inputs, the higher hit rates afforded by extensive aliasing has been found to be extremely beneficial even when it does carry with it a lower average precision [5].

3.5.2 Dealing with imprecision: disambiguation

Our data show that any keyword system capable of providing a high hit rate for unfamiliar users must let them use words of their own choice for objects. But, because of the inherent ambiguity of users’ own words, the system can never be sure it has correctly inferred the user’s referent; it can only make good guesses. The cost of an incorrect guess must therefore be considered. While users might be happy with a system’s mere guess about a book to look at, many commands are too consequential to be executed on guessed intent (e.g., "delete file") and so the inherent vagueness in words must be resolved.

There are two approaches: either make the user memorize precise system meanings, or have the user and system interact to identify the precise referent. The latter might possibly be done in many ways, including interpretation of multiterm Boolean expressions[6], formal query languages or "natural language" understanding, although none of these has demonstrated notable success to date. A simpler approach, which our data lets us model, is to return a set of choices to the user whenever there is ambiguity. If we assume that in the restricted context of a particular application, the user can recognize the correct interpretation on sight, there is probably no faster route to resolution. As we will show below, with unlimited aliases the number of alternatives to be returned for user selection will usually be small. Alternatives can be ordered by frequency, the most likely objects first. Nevertheless, correct recognition by the user requires adequate description of the meaning or content of each choice, perhaps including more precise terms, examples, etc. It must be realized that this too is a highly error prone communication process[2].

One question raised by the unlimited alias proposal is whether people will acquire undesirable habits if the system is so tolerant. We think this problem is self limiting. As mentioned, in command execution (though not necessarily in information retrieval) spontaneous entries will often require disambiguation procedures. Since such procedures are inherently costly to users, there is incentive for them to move toward more efficient language. This suggests using the unlimited alias system only as an index into a help facility. When users invoke it with their own words, the system would pick its best guesses, and present them in a menu. Each guess would be labeled by some "standard" access terminology and be accompanied by a description of the standard referent. Actual execution would always be via the standard "name," thereby encouraging the learning of precise terms required to take the disambiguation process out of the loop.

[6] Note that in terms of a system correctly recognizing untutored users' terms, boolean conjunctions *multiply* the single-term problem addressed in this paper. For example, indexer and retriever might agree to call large things, "big", with probability ˜0.1, and circular things, "round", with probability ˜0.2. Then, assuming independence, the boolean combination, "big and round" will retrieve only 0.02 of the desired large, circular things.

3.5.3 Performance of unlimited aliasing systems

We wish to know how well unlimited aliasing can work in principle and what can be expected in practice.

First, how good would performance be given an infinite amount of data? By definition the system would recognize all user words. However, it might have to make several guesses about the intended object. Assume the system asked the user to verify one object at a time, and the user was only satisfied by exactly one object in the database. Then, for our data sets, the median number of guesses before success in each disambiguation dialogue would be approximately as follows (the number in parentheses is a reminder of how many objects there were in the domain and is useful to help judge the magnitude of the accomplishment).

Results Table 6
Median number of system object guesses for success

	Editor-5	Editor-25	Decoder	Common Objects	Classifieds	Recipe Keywords
Guesses	1	3	1	1	1	4
Objects in domain	(5)	(25)	(12)	(50)	(64)	(188)

In real life, we can never have complete data on word-object usage. Incomplete data can have several detrimental effects, the most important being that the system might not know a user’s word at all, or only have it listed with wrong objects. This would lead to a complete miss, of the sort found very frequently in armchair naming methods.

The seriousness of this problem varied from domain to domain. In the decoder data, even after 100 subjects were asked about each object, a new person would have had a 28% chance of proposing a completely new term. In contrast, after only 24 people had proposed keywords for a recipe, there was less than one chance in 50 that the next person would come up with a new term.

There is no single number characterizing how overall performance would suffer from incomplete data; it would depend on how much data was collected and the variability of naming in the given domain. We estimated how well a system could do in our domains with half the amount of data we originally collected. At the median number of guesses needed with infinite data, performance was degraded by approximately one third when restricted to half our available data: performance was cut in half for Classifieds using only 8 subjects’ data, cut by a third for the Common Objects using 178 subjects’ data, and cut by only 10% for the Editor-5 data using 24 subjects’ data.

4. Summary and Discussion

We have been studying vocabulary problems in using unfamiliar computer applications. We can summarize the findings in a few major points. First, a single access term (e.g., a "name") chosen by a single designer will provide very poor access (10-20% hit rates). Second, a single empirically optimized term is much better; and will work as well as 2-3 armchair aliases together. Third, to achieve really good performance, very many aliases are needed. An empirically based, frequency weighted "unlimited aliases" system can often produce 50% to 100% hit rates in its first three guesses to untutored queries, depending on the domain and the amount of data collected. In the limit, the only barrier is precision, and our data show that frequency data on the possible objects intended by a given term will usually provide good ambiguity resolution.

This brings us to the conclusion that reasonable access can be provided for the untutored, but only if an extensive table of word usage behavior is compiled. Such a solution is technically simple, but potentially tedious. We note that to some extent the collection of multiple aliases will arise in any well done iterative interface design. Good, et al. [6] provide such an example. They report that alias collection accounted for the largest share of the dramatic interface performance improvements resulting from a careful iterative design process with extensive user testing.

Iterative design of interfaces is very useful for many reasons, but our analyses suggest the vocabulary problem can be attacked separately. But what can be done about the tedium and cost? Fortunately there appear to be several attractive alternatives. A "quick and dirty" approach is to get a fair number, say 4 to 8 representative users to supply a fair number, say 3 to 6, terms apiece for each object. An experiment by Gomez and Lochbaum[5] with a small interactive search system obtained the predicted four-fold increase in user’s ability to find a desired recipe using this approach. This technique is most likely to be cost-effective for databases of moderate size that are expected to remain stable over long periods and be used infrequently by many people.

A more familiar approach for textual objects is to extract multiple access terms automatically from the content. The recipe texts contain roughly 80% of the keywords selected by users, and in an experiment to be reported elsewhere, Gomez, Lochbaum and Landauer found full-text indexing to be roughly equivalent to extensive alias "harvesting" from experts for the recipe database.

A third, exciting possibility is to construct unlimited alias indices adaptively, on site, in use. Such adaptive indexing methods are discussed elsewhere[3] but the idea is simple and has some partial precursors in the literature [9] [10]. Basically an adaptive indexing system begins with an index obtained in an ordinary way (for example, by armchair naming). Users try their own words to access desired objects, and, on their first few tries, usually fail. But sometimes they will eventually find something they want. With user concurrence, the initially unsuccessful words are added to the system's usage table. The next time someone seeks the object, the index is more knowledgeable. Adaptive indexes collect the needed data relatively painlessly, from the right users under authentic conditions. They also start to pay off rapidly, since they automatically tend focus data collection on the most sought-after objects. Trials of such systems show that they produce the kind of large performance improvements predicted in this paper for systems with massive alias lists.

Thus we can augment our earlier conclusion to say that not only is drastic improvement needed and possible, but also practicable.

In closing, we would like to comment briefly on the research strategy that was employed in this work. Typically, human-computer interaction research has involved the comparison of two or more particular systems or features, either by model-based analysis (e.g., Card, Moran and Newell [1]), by a one-time "contest", (e.g., Roberts and Moran[11]) or by successive iterative test (e.g., Good, et al. [6]). This approach is valuable in increasing the speed and accuracy with which the field can choose good examples to follow. But, because of the complexity of most systems, it is limited in the generality of its conclusions.

By contrast, the approach exemplified here begins with what we call a failure analysis. In particular, we look for failures of humans to obtain desired results in interactions with computers. In the present case, we found that people often enter words that fail to be correctly recognized by systems. We then collect experimental or observational data, do simulations and model building -- whatever is necessary to understand what the human can and will do in the situation, and what is needed on the part of the machine in order that the combination of human and machine can succeed more often. In the present case, we discovered unexpectedly large and pervasive variability in spontaneous term application, and learned that systems need to recognize a very rich variety of aliases. If the emerging machine requirements can be realized practically, the next step to invention or improved design is straightforward. In the present case, many avenues for interface improvements are apparent.

It is essential to stress, that in this approach, inventions and design improvements come from basic understanding of performance problems and human behavior, not from analysis or tests of alternative designs. The extreme variability of word selection by humans is a fundamental fact of human behavior found by studying a performance problem. Knowledge of this fact can lead to better system design in a wide range of applications, not just in the choice of one system over another.

REFERENCES

1. Card, S., Moran, T.P. and Newell, A. *The Psychology of Human-Computer Interaction.* Hillsdale, NJ: Lawerence Erlbaum Associates. 1983.

2. Dumais, S.T. and Landauer, T.K., Describing categories of objects for menu retrieval systems. *Behavior Research Methods, Instruments, & Computers, 16,* 2 (Apr. 1984), 242-248.

3. Furnas, G.W., "Experience with an adaptive indexing scheme," *Human Factors in Computer Systems, CHI'85 Proceedings.* Conference held in San Francisco, CA, April 15-18, 1985, 131-135.

4. Furnas, G.W., Landauer, T.K., Gomez, L.M., and Dumais, S.T., Statistical semantics: Analysis of the potential performance of key-word information systems. *Bell System Technical Journal, 62,* 6 (Jul.-Aug. 1983), 1753-1806.

5. Gomez, L.M. and Lochbaum, C.C., People can retrieve more objects with enriched key-word vocabularies. But is there a human performance cost? In B. Shackel (Ed.) *Human-Computer Interaction - Interact '84,* North-Holland, Amsterdam, 257-261.

6. Good, M.D., Whiteside, J.A. Wixon, D.R., and Jones, S.J., Building a user-derived interface, *Commun. ACM, 27,* 10 (Oct. 1984), 1032-1043.

7. Herdan, G., *Type Token Mathematics: A Textbook of Mathematical Linguistics*, S-Gravenhage, Mouton, 1960.

8. Landauer, T.K., Gallotti, K., and Hartwell, S., Natural command names and initial learning: A study of text editing terms, *Commun. ACM, 26,* 7 (Jul. 1983), 495-503.

9. Reisner, P. Construction of a growing thesaurus by conversational interaction in a man-machine system. *Proceedings of the American Documentation Institute*, 26th Annual Meeting, Chicago, Ill. October 1963.

10. Reisner, P. Evaluation of a 'Growing Thesaurus'. Research Paper RC-1662, August 9, 1966, IBM Watson Research Center, Yorktown Heights, New York.

11. Roberts, T.L. and Moran, T.P., The evaluation of text editors: Methodology and empirical results. *Commun. ACM, 26,* 4 (Apr.1983), 265-283.

12. Spark-Jones, K. A Statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation 28,* 1 (Mar. 1972), 11-21.

13. Whalen, T. and Latremouille, S., The effectiveness of a tree-structured index when the existence of information is uncertain. *Teledon Behavioral Research 2: The Design of Videotex Tree Indices,* Ottawa, Canada: Department of Communications, (May 1981),pp. 3-12.

14. Zipf, G.K., *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology.* Addison-Wesley, Reading, MA, 1949.

The Vocabulary Problem in Human-System Communication: an Analysis and a Solution

G. W. Furnas
T. K. Landauer
L. M. Gomez
S. T. Dumais

Bell Communications Research
E

ABSTRACT

In almost all computer applications, users must enter correct words for the desired objects or actions. For success without extensive training, or in first-tries for new targets, the system must recognize terms that will be chosen spontaneously. We studied spontaneous word choice for objects in five application-related domains and found the variability to be surprisingly large. In every case two people favored the same term with probability <0.20. Simulations show how this fundamental property of language limits the success of various design methodologies for vocabulary-driven interaction. For example, the popular approach in which access is via one designer's favorite single word will result in 80-90% failure rates in many common situations. An optimal strategy, unlimited aliasing, is derived and shown to be capable of several-fold improvements.

SUMMARY

Many, many alternative access words are needed for users to get what they want from large and complex systems.

CR Categories and Subject Descriptors:
H.1.2 [Models and Principles]: User/Machine Systems -- *human factors, human information processing;*
H.3.1 [Information storage and retrieval]: Content Analysis and Indexing -- *Indexing Methods, Linguistic Processing, Thesauruses*

General term: Human Factors

Additional Keywords and Phrases: vocabulary, keyword, human-computer interaction, human-computer communication, repeat rate, Zipf law

NOTE: A much longer, highly technical version of some early parts of this work appeared in [3].

Authors' Address: Bell Communications Research, Inc., 435 South Street, Morristown, New Jersey, 07960

FIGURE CAPTIONS

Figure 1. To demonstrate how rarely two people will agree on what to call things, ask one other colleague to try it too and compare your responses.

Figure 2. Plot of how many different names (or "aliases") are needed for each object to account for a given percentage of success. The exact values are not known because of statistical estimation problems, so ranges are indicated by dashed lines.

Table 1 - Word-object data

(a) Sample data from the text-editing study

	Objects					
Words	*"insert"*	*"delete"*	*"replace"*	*"move"*	*"transpose"*	...
change	30	22	60	30	41	
remove	0	21	12	17	5	
spell	4	14	13	12	10	
reverse	0	0	0	0	27	...
leave	10	0	0	1	0	
make into	0	4	0	0	1	
.	.	.	.	.	.	
.	.	.	.	.	.	
.	.	.	.	.	.	

(b) Sample data from the common object study

	Objects						
Words	*"calculator*	*"nectarine"*	*"Lucille Ball"*	*"pear"*	*"raisin"*	*"robin"*	...
machine	4	0	0	0	0	0	
green	0	0	0	7	0	0	
bird	0	0	0	0	0	21	...
fruit	0	12	0	19	1	0	
red	0	0	8	0	0	7	
female	0	0	2	0	0	0	
.	.	.	.	.	.	.	
.	.	.	.	.	.	.	
.	.	.	.	.	.	.	

On a piece of paper write the name you would give to a program that tells about interesting activities occurring in some major metropolitan area (e.g., this program would tell you what is interesting to do on Friday or Saturday night). Make the name 10 characters or less.

Try to think of a name that will be as obvious as possible,
one that other people would think of.

Figure 1

The Vocabulary Problem in Human-System Communication

G. W. Furnas
T. K. Landauer
L. M. Gomez
S. T. Dumais

Bell Communications Research
435 South Street
Morristown, NJ 07960