The New C Standard (Sentence 782)

An Economic and Cultural Commentary

Derek M. Jones
derek@knosof.co.uk

Acknowledgments

The New C Standard: An economic and cultural commentary -1

Commentary

Thanks to Sean Corfield and later Gavin Halliday for many interesting discussions on C90 implementation. Also Clive Feather, the UK C panel, and the members of WG14 were the source of many insights.

Clive Feather reviewed most of the material in this book. Fred Tydeman reviewed the floating-point material in all subsections. Frank Griswold provided a detailed review of over half of the C++ material. Stephen Parker review an very early draft of some of the coding guidelines.

Most of the work on the scripts and style sheets/macros use for the layout was done by Vic Kirk. Thanks to the authors of TeXlive, grap, pic, graphviz, and a variety of 'nix based tools.

Marilyn Rash (rrocean@shore.net) copyedited 75% of the material.

Thanks to the librarians of Reading, Surrey, and Warwick Universities for providing access to their collections of Journals. Thanks to all those people who made their papers available online (found via Altavista and later Google).

Conventions

0 This is a sentence from the C99 standard, the number on the inside margin (it would be in a bound book) is the sentence number and this wording has been ~~deleted~~added by the response to a DR.

information defined here

Commentary

This is some insightful commentary on the above sentence. We might also say something relating to this issue in another sentence (see sentence number and reference heading in the outside marginwould be in a bound book).

[] another sentence

The term *blah* is delimited with single-quotes to highlight the fact that it is defined as a term.

This is a quote from the Rationale document produced by the C Committee to put a thoughtful spin on the wording in the standard.

Rationale

C90

This section deals with the C90 version of the standard. Specifically, how it differs from the C99 version of the above sentence. These sections only appear if there is a semantic difference (in some cases the words may have changed slightly, leaving the meaning unchanged).

This is the text of a DR (defect report) submitted to the ISO C Standard committee.

Response

The committees response to this DR is that this question is worth repeating at this point in the book.

DR #987

This is where we point out what the difference, is any (note the change bar), and what the developer might do, if anything, about it.

C++

This is a sentence from the C++ standard specifying behavior that is different from the above C99 sentence. The 1.1p1 in the outside margin is the clause and paragraph number of this quote in the C++ Standard.

1.1p1

This is where we point out what the difference is, and what the developer might do, if anything, about it. You believed the hype that the two languages are compatible? Get real!

Other Languages

Developers are unlikely to spend their entire professional life using a single language. This section sometimes gives a brief comparison between the C way of doing things and other languages.

We vote against the adoption of the proposed new COBOL standard because we have lost some of our source code and don't know whether the requirements in the proposed new standard would invalidate this source.

Comment received during balloting

Common Implementations

Discussion of how implementations handle the above sentence. For instance, only processors with 17 bit integers can implement this requirement fully (note the text in the outside columnleft or flush right to the edge of the pagea heading that can be referenced from elsewhere). GCC has extensions to support 16 bit processors in this area (the text in the outside margin is pushed towards the outside of the page, indicating that this is where a particular issue is discussed; the text appearing in a smaller point size is a reference to material appearing elsewhere {the number is the C sentence number}).

processors 17 bit

[] translated invalid program

This is a quote from the document referenced in the outside sidebar.

Coding Guidelines

General musings on how developers use constructs associated with the above sentence. Some of these sections recommend that a particular form of the construct described in the above sentence not be used.

[1] Do it this way and save money.

DEV A possible deviation from the guideline, for a described special case.

REV Something to look out for during a code review. Perhaps a issue that requires a trade off among different issues, or that cannot be automated.

Example

An example, in source code of the above sentence.

The examples in this book are generally intended to illustrate some corner of the language. As a general rule it is considered good practice for authors to give examples that readers should follow. Unless stated otherwise, the examples in this book always break this rule.

```
1  struct {float mem;} main(void)
2  {
3  int blah;  /* The /* form of commenting describes the C behavior   */
4             // The // form of commenting describes the C++ behavior
5  }
```

Usage

A graph or table showing the number of occurrences the constructs specified by the above C sentence occur in source code.

identifier
syntax

782

identifier:
identifier-nondigit
identifier identifier-nondigit
identifier digit
identifier-nondigit:
nondigit
universal-character-name
other implementation-defined characters
nondigit: one of
_ a b c d e f g h i j k l m
n o p q r s t u v w x y z
A B C D E F G H I J K L M
N O P Q R S T U V W X Y Z
digit: one of
0 1 2 3 4 5 6 7 8 9

Commentary

From the developer's point of view identifiers are the most important tokens in the source code. The reasons for this are discussed in the Coding guidelines section that follows.

C90

Support for *universal-character-name* and "other implementation-defined characters" is new in C99.

C++

The C++ Standard uses the term *nondigit* to denote an *identifier-nondigit*. The C++ Standard does not specify the use of *other implementation-defined characters*. This is because such characters will have been replaced in translation phase 1 and not be visible here.

translation phase 1

Other Languages

Some languages do not support the use of underscore, _, in identifiers. There is a growing interest from the users of different computer languages in having support for *universal-character-name* characters in identifiers. But few languages have gotten around to doing anything about it yet. What most other languages call operators can appear in identifiers in Scheme (but not as the first character). Java was the first well-known language to support *universal-character-name* characters in identifiers.

Common Implementations

Some implementations support the use of the $ character in identifiers.

Coding Guidelines

identifier introduction

1 Introduction

1.1 Overview

This coding guideline section contains an extended discussion on the issues involved with reader's use of identifier names, or spellings.[1] It also provides some recommendations that aim to prevent mistakes from being made in their usage.

Identifiers are the most important token in the visible source code from the program comprehension perspective. They are also the most common token (29% of the visible tokens in the `.c` files, with comma being the second most common at 9.5%), and they represent approximately 40% of all non-white-space characters in the visible source (comments representing 31% of the characters in the `.c` files).

From the developer's point of view, an identifier's spelling has the ability to represent another source of information created by the semantic associations it triggers in their mind. Developers use identifier spellings both as an indexing system (developers often navigate their way around source using identifiers) and as an aid to comprehending source code. From the translators point of view, identifiers are simply a meaningless sequence of characters that occur during the early stages of processing a source file. (The only operation it needs to be able to perform on them is matching identifiers that share the same spellings.)

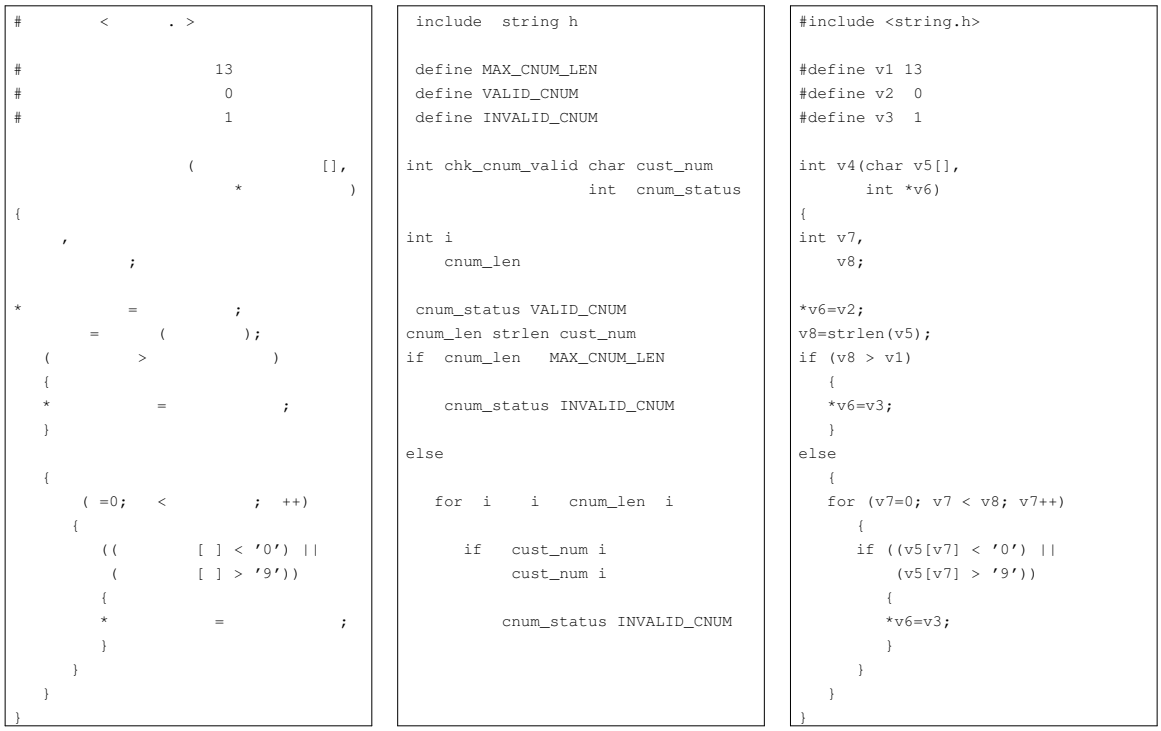
identifier cue for recall

The information provided by identifier names can operate at all levels of source code construct, from providing helpful clues about the information represented in objects at the level of C expressions (Figure 1) to a means of encapsulating and giving context to a series of statements and declaration in a function definition. An example of the latter is provided by a study by Bransford and Johnson[35] who read subjects the following passage (having told them they would have to rate their comprehension of it and would be tested on its contents).

Bransford and Johnson[35]

> *The procedure is really quite simple. First you arrange things into different groups depending on their makeup. Of course, one pile may be sufficient depending on how much there is to do. If you have to go somewhere else due to lack of facilities that is the next step, otherwise you are pretty well set. It is important not to overdo any*

[1] Common usage is for the character sequence denoting an identifier to be called its *name*; these coding guidelines often use the term *spelling* to prevent possible confusion.



```
#        <      . >

#                    13
#                     0
#                     1

                  (             [],
                       *           )
{
     ,
            ;

*           =          ;
        =      (        );
   (          >             )
   {
   *           =            ;
   }

   {
       ( =0;   <          ;  ++)
      {
         ((        [ ] < '0') ||
          (        [ ] > '9'))
         {
         *           =            ;
         }
      }
   }
}
```

```
 include  string h

 define MAX_CNUM_LEN
 define VALID_CNUM
 define INVALID_CNUM

int chk_cnum_valid char cust_num
                    int  cnum_status

int i
    cnum_len

 cnum_status VALID_CNUM
cnum_len strlen cust_num
if  cnum_len   MAX_CNUM_LEN

    cnum_status INVALID_CNUM

else

   for  i    i   cnum_len  i

      if   cust_num i
           cust_num i

          cnum_status INVALID_CNUM
```

```
#include <string.h>

#define v1 13
#define v2  0
#define v3  1

int v4(char v5[],
       int *v6)
{
int v7,
    v8;

*v6=v2;
v8=strlen(v5);
if (v8 > v1)
   {
   *v6=v3;
   }
else
   {
   for (v7=0; v7 < v8; v7++)
      {
      if ((v5[v7] < '0') ||
          (v5[v7] > '9'))
         {
         *v6=v3;
         }
      }
   }
}
```

Figure 1: The same program visually presented in three different ways; illustrating how a reader's existing knowledge of words can provide a significant benefit in comprehending source code. By comparison, all the other tokens combined provide relatively little information. Based on an example from Laitinen.[187]

particular endeavor. That is, it is better to do too few things at once than too many. In the short run this may not seem important, but complications from doing too many can easily arise. A mistake can be expensive as well. The manipulation of the appropriate mechanisms should be self-explanatory, and we need not dwell on it here. At first the whole procedure will seem complicated. Soon, however, it will become just another facet of life. It is difficult to foresee any end to this task in the immediate future, but then one never can tell.

Table 1: Mean comprehension rating and mean number of ideas recalled from passage (standard deviation is given in parentheses). Adapted from Bransford and Johnson.[35]

	No Topic Given	Topic Given After	Topic Given Before	Maximum Score
Comprehension	2.29 (0.22)	2.12 (0.26)	4.50 (0.49)	7
Recall	2.82 (0.60)	2.65 (0.53)	5.83 (0.49)	18

The results (Table 1) show that subjects recalled over twice as much information if they were given a meaningful phrase (the topic) before hearing the passage. The topic of the passage describes washing clothes

The basis for this discussion is human language and the cultural conventions that go with its usage. People spend a large percentage of their waking day, from an early age, using this language (in spoken and written form). The result of this extensive experience is that individuals become tuned to the commonly occurring sound and character patterns they encounter (this is what enables them to process such material automatically without apparent effort). This experience also results in an extensive semantic network of associations for the words of a language being created in their head. By comparison, experience reading source code pales into insignificance.

[] reading practice

[] automatization
[] semantic networks

These coding guidelines do not seek to change the habits formed as a result of this communication experience using natural language, but rather to recognize and make use of them. While C source code is a written, not a spoken language, developers' primary experience is with a spoken language that also has a written form.

The primary factor affecting the performance of a person's character sequence handling ability appears to be the characteristics of their native language (which in turn appears to have been tuned to the operating characteristics of the brains of its speakers). This coding guideline discussion makes the assumption that developers will attempt to process C language identifiers like the words and phrases of their native language (i.e., the characteristics of a developer's native language are the most significant factor in their processing of identifiers). The operating characteristics of the brain also affect performance (e.g., short-term memory is primarily sound based and information lookup is via spreading activation).

There are too many permutations and combinations of possible developer experiences for it to be possible to make general recommendations on how to optimize the selection of identifier spellings. A coding guideline recommending that identifier spellings match the characteristics, spoken as well as written, and conventions (e.g., word order) of the developers' native language is not considered to be worthwhile because it is a practice that developers appear to already, implicitly follow. (Some suggestions on spelling usage are given.) However, it is possible to make guideline recommendations about the use of identifier spellings that are likely to be a cause of problems. These recommendations are essentially filters of spellings that have already been chosen.

identifier suggestions
identifier filtering spellings

1.2 Primary identifier spelling issues

identifier primary spelling issues

There are several ways of dividing up the discussion on identifier spelling issues (see Table 2). The headings under which the issues are grouped is a developer-oriented ones (the expected readership for this book rather than a psychological or linguistic one). The following are the primary issue headings used:

- *Memorability*. This includes recalling the spelling of an identifier (given some semantic information associated with it), recognizing an identifier from its spelling, and recalling the information associated with an identifier (given its spelling). For instance, what is the name of the object used to hold the current line count, or what information does the object `zip_zap` represent?
- *Confusability*. Any two different identifier spellings will have some degree of commonality. The greater the number of features different identifiers have in common, the greater the probability that a reader will confuse one of them for the other. Minimizing the probability of confusing one identifier with a different one is the ideal, but these coding guidelines attempt have the simpler aim of preventing mutual confusability between two identifiers exceeding a specified level,
- *Usability*. Identifier spellings need to be considered in the context in which they are used. The memorability and confusability discussion treats individual identifiers as the subject of interest, while usability treats identifiers as components of a larger whole (e.g., an expression). Usability factors include the cognitive resources needed to process an identifier and the semantic associations they evoke, all in the context in which they occur in the visible source (a more immediate example might be the impact of its length on code layout). Different usability factors are likely to place different demands on the choice of identifier spelling, requiring trade-offs to be made.

expression visual layout

Table 2: Break down of issues considered applicable to selecting an identifier spelling.

	Visual	Acoustic	Semantic	Miscellaneous
Memory	Idetic memory	Working memory is sound based	Proper names, LTM is semantic based	spelling, cognitive studies, Learning
Confusability	Letter and word shape	Sounds like	Categories, metaphor	Sequence comparison
Usability	Careful reading, visual search	Working memory limits, pronounceability	interpersonal communication, abbreviations	Cognitive resources, typing

A spelling that, for a particular identifier, maximizes memorability and usability while minimizing confusability may be achievable, but it is likely that trade-offs will need to be made. For instance, human short-term memory capacity limits suggest that the duration of spoken forms of an identifier's spelling, appearing as operands in an expression, be minimized. However, identifiers that contain several words (increased speaking time), or rarely used words (probably longer words taking longer to speak), are likely to invoke more semantic associations in the readers mind (perhaps reducing the total effort needed to comprehend the source compared to an identifier having a shorter spoken form).

[] memory developer

If asked, developers will often describe an identifier spelling as being either *good* or *bad*. This coding guideline subsection does not measure the quality of an identifier's spelling in isolation, but relative to the other identifiers in a program's source code.

1.2.1 Reader language and culture

developer language and culture

During the lifetime of a program, its source code will often be worked on by developers having different first languages (their native, or mother tongue). While many developers communicate using English, it is not always their first language. It is likely that there are native speakers of every major human language writing C source code.

Of the 3,000 to 6,000 languages spoken on Earth today, only 12 are spoken by 100 million or more people (Table 3). The availability of cheaper labour outside of the industrialized nations is slowly shifting developers' native language away from those nations' languages to Mandarin Chinese, Hindi/Urdu, and Russian.

If English was good enough for Jesus, it is good enough for me (attributed to various U.S. politicians).

Table 3: Estimates of the number of speakers each language (figures include both native and nonnative speakers of the language; adapted from Ethnologue volume I, SIL International). Note: Hindi and Urdu are essentially the same language, Hindustani. As the official language of Pakistan, it is written right-to-left in a modified Arabic script and called Urdu (106 million speakers). As the official language of India, it is written left-to-right in the Devanagari script and called Hindi (469 million speakers).

Rank	Language	Speakers (millions)	Writing direction	Preferred word order
1	Mandarin Chinese	1,075	left-to-right also top-down	SVO
2	Hindi/Urdu	575	see note	see note
3	English	514	left-to-right	SVO
4	Spanish	425	left-to-right	SVO
5	Russian	275	left-to-right	SVO
6	Arabic	256	right-to-left	VSO
7	Bengali	215	left-to-right	SOV
8	Portuguese	194	left-to-right	SVO
9	Malay/Indonesian	176	left-to-right	SVO
10	French	129	left-to-right	SVO
11	German	128	left-to-right	SOV
12	Japanese	126	left-to-right	SOV

If, as claimed here, the characteristics of a developer's native language are the most significant factor in their processing of identifiers, then a developer's first language should be a primary factor in this discussion. However, most of the relevant studies that have been performed used native-English speakers as subjects.[2] Consequently, it is not possible to reliably make any claims about the accuracy of applying existing models of visual word processing to non-English languages.

The solution adopted here is to attempt to be natural-language independent, while recognizing that most of the studies whose results are quoted used native-English speakers. Readers need to bear in mind that it is likely that some of the concerns discussed do not apply to other languages and that other languages will have concerns that are not discussed.

1.3 How do developers interact with identifiers?

identifier developer interaction

The reasons for looking at source code do not always require that it be read like a book. Based on the various reasons developers have for looking at source the following list of identifier-specific interactions are

[] reading kinds of

[2] So researchers have told your author, who, being an English monoglot, has no choice but to believe them.

considered:

- When quickly skimming the source to get a general idea of what it does, identifier names should suggest to the viewer, without requiring significant effort, what they are intended to denote.
- When searching the source, identifiers should not disrupt the flow (e.g., by being extremely long or easily confused with other identifiers that are likely to be seen).
- When performing a detailed code reading, identifiers are part of a larger whole and their names should not get in the way of developers' appreciation of the larger picture (e.g., by requiring disproportionate cognitive resources).

trust based usage

- Trust based usage. In some situations readers extract what they consider to be sufficiently reliable information about an identifier from its spelling or the context in which it is referenced; they do not invest in obtaining more reliable information (e.g., by, locating and reading the identifiers' declaration).

Developers rarely interact with isolated identifiers (a function call with no arguments might be considered to be one such case). For instance, within an expression an identifier is often paired with another identifier (as the operand of a binary operator) and a declaration often declares a list of identifiers (which may, or may not, have associations with each other).

However well selected an identifier spelling might be, it cannot be expected to change the way a reader chooses to read the source. For instance, a reader might keep identifier information in working memory, repeatedly looking at its definition to refresh the information; rather like a person repeatedly looking at their watch because they continually perform some action that causes them to forget the time and don't invest (perhaps because of an unconscious cost/benefit analysis) the cognitive resources needed to better integrate the time into their current situation.

Introducing a new identifier spelling will rarely causes the spelling of any other identifier in the source to be changed. While the words of natural languages, in spoken and written form, evolve over years, experience shows that the spelling of identifiers within existing source code rarely changes. There is no perceived cost/benefit driving a need to make changes.

cost/accuracy trade-off

An assumption that underlies the coding guideline discussions in this book is that developers implicitly, and perhaps explicitly, make cost/accuracy trade-offs when working with source code. These trade-offs also occur in their interaction with identifiers.

word visual recognition

1.4 Visual word recognition

This section briefly summarizes those factors that are known to affect visual word recognition and some of the models of human word recognition that have been proposed. A word is said to be recognized when its representation is uniquely accessed in the reader's lexicon. Some of the material in this subsection is based on chapter 6 of *The Psychology of Language* by T. Harley.[134]

Reading is a recent (last few thousand years) development in human history. Widespread literacy is even more recent (under 100 years). There has been insufficient time for the impact of comparative reading skills to have had any impact on our evolution, assuming that it has any impact. (It is not known if there is any correlation between reading skill and likelihood of passing on genes to future generation.) Without evolutionary pressure to create specialized visual word-recognition systems, the human word-recognition system must make use of cognitive processes designed for other purposes. Studies suggest that word recognition is distinct from object recognition and specialized processes, such as face recognition. A model that might be said to mimic the letter- and word-recognition processes in the brain is the Interactive Activation Model.[215]

The psychology studies that include the use of character sequences (in most cases denoting words) are intended to uncover some aspect of the workings of the human mind. While the tasks that subjects are asked to perform are not directly related to source code comprehension, in some cases, it is possible to draw parallels. The commonly used tasks in the studies discussed here include the following:

- *The naming task.* Here subjects are presented with a word and the time taken to name that word is measured. This involves additional cognitive factors that do not occur during silent reading (e.g., controlling the muscles that produce sounds). naming task
- *The lexical decision task.* Here subjects are asked to indicate, usually by pressing the appropriate button, whether a sequence of letters is a word or nonword (where a word is a letter sequence that is the accepted representation of a spoken word in their native language). lexical decision task
- *The semantic categorization task.* Here subjects are presented with a word and asked to make a semantic decision (e.g., "is *apple* a fruit or a make of a car?"). semantic categorization task

The following is a list of those factors that have been found to have an effect on visual word recognition. Studies[3, 141] investigating the interaction between these factors have found that there are a variety of behaviors, including additive behavior and parallel operation (such as the Stroop effect). [] stroop effect

- *Age of acquisition*. Words learned early in life are named more quickly and accurately than those learned later.[331] Age of acquisition interacts with frequency in that children tend to learn the more common words first, although there are some exceptions (e.g., *giant* is a low-frequency word that is learned early). age of acquisition
- *Contextual variability*. Some words tend to only occur in certain contexts (low-contextual variability), while others occur in many different contexts (high-contextual variability). For instance, in a study by Steyvers and Malmberg[293] the words *atom* and *afternoon* occurred equally often; however, *atom* occurred in 354 different text samples while *afternoon* occurred in 1,025. This study found that words having high-contextual variability were more difficult to recognize than those having low-contextual variability (for the same total frequency of occurrence).
- *Form-based priming* (also known as *orthographic priming*). The form of a word might be thought to have a priming effect; for instance, *CONTRAST* shares the same initial six letters with *CONTRACT*. Studies have failed to find any measurable effects.
- *Illusory conjunctions*. These occur when words are presented almost simultaneously, as might happen when a developer is repeatedly paging through source on a display device; for instance, the letter sequences *psychment* and *departology* being read as *psychology* and *department*. illusory conjunctions
- *Length effects*. There are several ways of measuring the length of a word; they tend to correlate with each other (e.g., the number of characters vs. number of syllables). Studies have shown that there is some effect on naming for words with five or more letters. Naming time also increases as the number of syllables in a word increases (also true for naming pictures of objects and numbers with more syllables). Some of this additional time includes preparing to voice the syllables.
- *Morphology*. The stem-only model of word storage[300] proposed that word stems are stored in memory, along with a list of rules for prefixes (e.g., *re* for performing something again) and suffixes (*ed* for the past tense), and their exceptions. The model requires that these affixes always be removed before lookup (of the stripped word). Recognition of words that look like they have a prefix (e.g., *interest*, *result*), but don't, has been found to take longer than words having no obvious prefix (e.g., *crucial*). Actual performance has been found to vary between different affixes. It is thought that failure to match the letter sequence without the prefix causes a reanalysis of the original word, which then succeeds. See Vannest[316] for an overview and recent experimental results. morphology identifier
- *Neighborhood effects*. It is possible to convert many words to another word by changing a single letter. Words that differ by a single letter are known as *orthographic neighbors*. Some words have many orthographic neighbors—*mine* has 29 (*pine*, *line*, *mane*, etc.)—while others have few. Both the *density* of orthographics neighbors (how many there are) and their relative frequency (if a neighbor occurs more or less frequently in written texts) can affect visual word recognition. The *spread* of neighborhood identifier

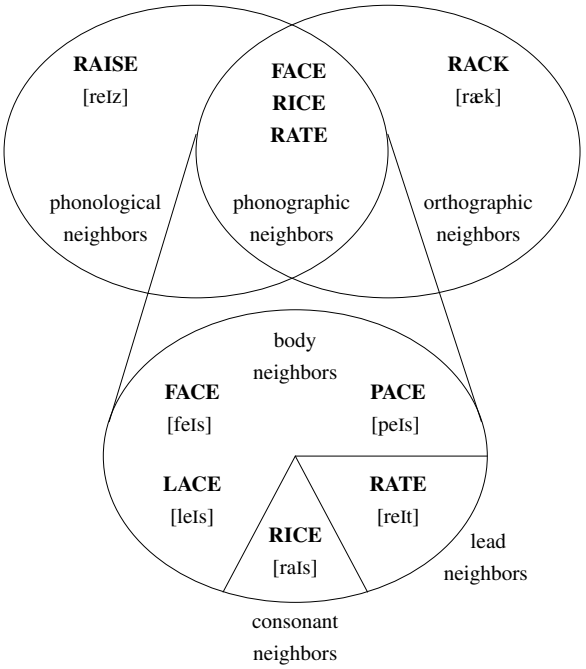


Figure 2: Example of the different kinds of lexical neighborhoods for the English word *RACE*. Adapted from Peereman and Content.[243]

the neighbors for a particular word is the number of different letter positions that can be changed to yield a neighbor (e.g., *clue* has a spread of two—*glue* and *club*). The rime of neighbors can also be important; see Andrews[8] for a review.

- *Nonword conversion effect*. A nonword is sometimes read as a word whose spelling it closely resembles.[257] This effect is often seen in a semantic priming context (e.g., when proofreading prose).
- *Other factors*. Some that have been suggested to have an effect on word recognition include meaningfulness, concreteness, emotionality, and pronounceability,

phonological neighborhood identifier

- *Phonological neighborhood*. Phonological neighborhood size has not been found to be a significant factor in processing of English words. However, the Japanese lexicon contains many homophones. For instance, there are many words pronounced as */kouen/* (i.e., park, lecture, support, etc.). To discriminate homophones, Japanese readers depend on orthographic information (different Kanji compounds). A study by Kawakami[168] showed that phonological neighborhood size affected subjects' lexical decision response time for words written in Katakana.
- *Proper names*. A number of recent studies[146] have suggested that the cognitive processing of various kinds of proper names (e.g., people's names and names of landmarks) is different from other word categories.

words English

- *Repetition priming*. A word is identified more rapidly, and more accurately, on its second and subsequent occurrences than on its first occurrence. Repetition priming interacts with frequency in that the effect is stronger for low-frequency words than high-frequency ones. It is also affected by the number of items intervening between occurrences. It has been found to decay smoothly over the first three items for words, and one item for nonwords to a stable long-term value.[217]

semantic priming

- *Semantic priming*. Recognition of a word is faster if it is immediately preceded by a word that has a semantically similar meaning;[253] for instance, *doctor* preceded by the word *nurse*. The extent to

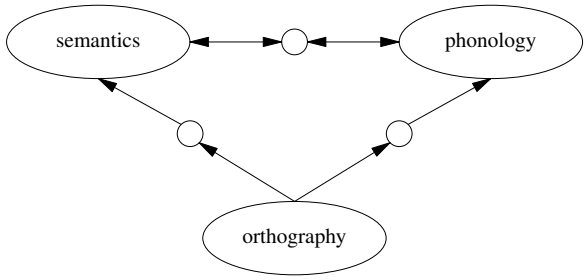


Figure 3: Triangle model of word recognition. There are two routes to both semantics and phonology, from orthography. Adapted from Harm.[135]

which priming occurs depends on the extent to which word pairs are related, the frequency of the words, the age of the person, and individual differences,

- *Sentence context*. The sentence "It is important to brush your teeth every" aids the recognition of the word *day*, the highly predictable ending, but not *year* which is not.
- *Syllable frequency*. There has been a great deal of argument on the role played by syllables in word recognition. Many of the empirical findings against the role of syllables have been in studies using English; however, English is a language that has ambiguous and ill-defined syllable boundaries. Other languages, such as Spanish, have well-defined syllable boundaries. A study by Álvarev, Carreiras, and de Vega[4] using Spanish-speaking subjects found that syllable frequency played a much bigger role in word recognition than in English. syllable frequency
- *Word frequency*. The number of times a person has been exposed to a word effects performance in a number of ways. High-frequency words tend to be recalled better, while low-frequency words tend to be better recognized (it is thought that this behavior may be caused by uncommon words having more distinctive features,[208,278] or because they occur in fewer contexts[293]). It has also been shown[142] that the attentional demands of a word-recognition task are greater for less frequent words. Accurate counts of the number of exposures an individual has had to a particular word are not available, so word-frequency measures are based on counts of their occurrence in large bodies of text. The so-called *Brown* corpus[181] is one well-known, and widely used, collection of English usage. (Although it is relatively small, one million words, by modern standards and its continued use has been questioned.[46]) The British National Corpus[195] (BNC) is more up-to-date (the second version was released in 2001) and contains more words (100 million words of spoken and written British English). word frequency
- *Word/nonword effects*. Known words are responded to faster than nonwords. Nonwords whose letter sequence does not follow the frequency distribution of the native language are rejected more slowly than nonwords that do. nonword effects

1.4.1 Models of word recognition

Word recognition models of

Several models have been proposed for describing how words are visually recognized.[160] One of the main issues has been whether orthography (letter sequences) are mapped directly to semantics, or whether they are first mapped to phonology (sound sequences) and from there to semantics. The following discussion uses the Triangle model.[135] (More encompassing models exist; for instance, the Dual Route Cascade model[63] is claimed by its authors to be the most successful of the existing computational models of reading. However, because C is not a spoken language the sophistication and complexity of these models is not required.)

By the time they start to learn to read, children have already built up a large vocabulary of word sounds (*phonology* $\Rightarrow$ *semantics*). This existing knowledge can be used when learning to read alphabetic scripts

such as English (see Siok and Fletcher[280] for a study involving logographic, Chinese, reading acquisition). They simply have to learn how to map letter sequences to the word sounds they already know (*orthography* ⇒ *phonology* ⇒ *semantics*). The direct mapping of sequences of letters to semantics (*orthography* ⇒ *semantics*) is much more difficult to learn. (This last statement is hotly contested by several psychologists and education experts who claim that children would benefit from being taught using the *orthography* ⇒ *semantics* based methods.)

The results of many studies are consistent with the common route, via phonology. However, there are studies, using experienced readers, which have found that in some cases a direct mapping from orthography to semantics occurs. A theory of visual word recognition cannot assume that one route is always used.

The model proposed by[135] is based on a neural network and an appropriate training set. The training set is crucial—it is what distinguishes the relative performance of one reader from another. A person with a college education will have read well over 20 million words by the time they graduate.[3]

words domain knowledge

Readers of different natural languages will have been trained on different sets of input. Even the content of courses taken at school can have an effect. A study by Gardner, Rothkopf, Lapan, and Lafferty[115] used 10 engineering, 10 nursing, and 10 law students as subjects. These subjects were asked to indicate whether a letter sequence was a word or a nonword. The words were drawn from a sample of high frequency words (more than 100 per million), medium-frequency (10–99 per million), low-frequency (less than 10 per million), and occupationally related engineering or medical words. The nonwords were created by rearranging letters of existing words while maintaining English rules of pronounceability and orthography.

The results showed engineering subjects could more quickly and accurately identify the words related to engineering (but not medicine). The nursing subjects could more quickly and accurately identify the words related to medicine (but not engineering). The law students showed no response differences for either group of occupationally related words. There were no response differences on identifying nonwords. The performance of the engineering and nursing students on their respective occupational words was almost as good as their performance on the medium-frequency words.

The Gardner et al. study shows that exposure to a particular domain of knowledge can affect a persons recognition performance for specialist words. Whether particular identifier spellings are encountered by individual developers sufficiently often, in C source code, for them to show a learning effect is not known.

identifier selecting spelling

2 Selecting an identifier spelling

2.1 Overview

This section discusses the developer-oriented factors involved in the selection of an identifier's spelling. The approach taken is to look at what developers actually do[4] rather than what your author or anybody else thinks they should do. Use of this approach should not be taken to imply that what developers actually do is any better than the alternatives that have been proposed. Given the lack of experimental evidence showing that the proposed alternatives live up to the claims made about them, there is no obvious justification for considering them.

Encoding information in an identifiers spelling is generally believed to reduce the effort needed to comprehend source code (by providing useful information to the reader).[5]

Some of attributes, information about which, developers often attempt to encode in an identifiers spelling include:

[3] A very conservative reading rate of 200 words per minute, for 30 minutes per day over a 10 years period.

[4] Some of the more unusual developer naming practices are more talked about than practiced. For instance, using the names of girl friends or football teams. In the visible form of the `.c` files 1.7% of identifier occurrences have the spelling of an English christian name. However, most of these (e.g., `val`, `max`, `mark`, etc.) have obvious alternative associations. Others require application domain knowledge (e.g., hardware devices: `lance`, floating point `nan`). This leaves a handful, under 0.01%. that may be actual uses of peoples names (e.g., `francis`, `stephen`, `terry`).

[5] The few studies that have investigated this belief have all used inexperienced subjects; there is no reliable experimental evidence to support this belief.

- *Information on what an identifier denotes.* This information may be application attributes (e.g., the number of characters to display on some output device) or internal program housekeeping attributes (e.g., a loop counter).
- *C language properties of an identifier.* For instance, what is its type, scope, linkage, and kind of identifier (e.g., macro, object, function, etc.).
- *Internal representation information.* What an object's type is, or where its storage is allocated.
- *Management-mandated information.* This may include the name of the file containing the identifier's declaration, the date an identifier was declared, or some indication of the development group that created it.

The encoded information may consist of what is considered to be more than one distinct character sequence. These distinct character sequences may be any combination of words, abbreviations, or acronyms. Joining together words is known as *compounding* and some of the rules used, primarily by native-English speakers, are discussed elsewhere. Studies of how people abbreviate words and the acronyms they create are also discussed elsewhere. Usability issues associated with encoding information about these attributes in an identifiers spelling is discussed elsewhere.

[] compound word
[] abbreviating identifier
[] identifier encoding usability

One conclusion to be drawn from the many studies discussed in subsequent sections is that optimal selection of identifier spelling is a complex issue, both theoretically and practically. Optimizing the memorability, confusability, and usability factors discussed earlier requires that the mutual interaction between all of the identifiers in a program's visible source code be taken into account, as well as their interaction with the reader's training and education. Ideally this optimization would be carried out over all the visible identifiers in a programs source code (mathematically this is a constraint-satisfaction problem). In practice not only is constraint satisfaction computationally prohibitive for all but the smallest programs, but adding a new identifier could result in the spellings of existing identifiers changing (because of mutual interaction), and different spelling could be needed for different readers, perhaps something that future development environments will support (e.g., to index different linguistic conventions).

optimal spelling identifier

The current knowledge of developer identifier-performance factors is not sufficient to reliably make coding guideline recommendations on how to select an identifier spelling (although some hints are made). However, enough is known about developer mistakes to be able to made some guideline recommendations on identifier spellings that should not be used.

This section treats creating an identifier spelling as a two-stage process, which iterates until one is selected:

1. *A list of candidates is enumerated.* This is one of the few opportunities for creative thinking when writing source code (unfortunately the creative ability of most developers rarely rises above the issue of how to indent code). The process of creating a list of candidates is discussed in the first subsection that follows.
2. *The candidate list is filtered.* If no identifiers remain, go to step 1. The factors controlling how this filtering is performed are discussed in the remaining subsections.

Some of the most influential ideas on how humans communicate meaning using language were proposed by Grice[129] and his maxims have been the starting point for much further research. An up-to-date, easier-to-follow discussion is provided by Clark,[57] while the issue of relevance is discussed in some detail by Sperber and Wilson.[286]

[] relevance

More detailed information on the theory and experimental results, which is only briefly mentioned in the succeeding subsections, is provided in the sections that follow this one.

2.2 Creating possible spellings

cost/accuracy trade-off

An assumption that underlies all coding guideline discussions in this book is that developers attempt (implicitly or explicitly) to minimize their own effort. Whether they seek to minimize immediate effort (needed to create the declaration and any associated reference that caused it to be created) or the perceived future effort of using that identifier is not known.

Zipf's law 782

Frequency of occurrence of words in spoken languages has been found to be approximately tuned so that shorter ones occur most often. However, from the point of view of resource minimization there is an important difference between words and identifiers. A word has the opportunity to evolve—its pronunciation can change or the concept it denotes can be replaced by another word. An identifier, once declared in the source, rarely has its spelling modified. The cognitive demands of a particular identifier are fixed at the time it is first used in the source (which may be a declaration, or a usage in some context soon followed by a declaration). This point of first usage is the only time when any attempt at resource minimization is likely to occur.

Developers typically decide on a spelling within a few seconds. Selecting identifier spellings is a creative process (one of the few really creative opportunities when working at the source code level) and generates a high cognitive load, something that many people try to avoid. Developers use a variety of cognitive load reducing decision strategies, which include spending little time on the activity.

When do developers create new identifiers? In some cases a new identifier is first used by a developer when its declaration is created. In other cases the first usage is when the identifier is referenced when an expression is created (with its declaration soon following). The semantic associations present in the developer's mind at the time an identifier spelling is selected, may not be the same as those present once more uses of the identifier have occurred (because additional uses may cause the relative importance given to the associated semantic attributes to change).

When a spelling for a new identifier is required a number of techniques can be employed to create one or more possibilities, including the following:

- *Waiting for one to pop into its creator head.* These are hopefully derived from semantic associations (from the attributes associated with the usage of the new identifier) indexing into an existing semantic network in the developers' head.

  semantic networks

- *Using an algorithm.* For instance, *template* spellings that are used for particular cases (e.g., using `i` or a name ending in `index` for a loop variable), or applying company/development group conventions (discussed elsewhere).

  loop control variable identifier other guideline documents

- *Basing the spelling on that of the spellings of existing identifiers* with which the new identifier has some kind of association. For instance, the identifiers may all be enumeration constants or structure members in the same type definition, or they may be function or macro names performing similar operations. Some of the issues (e.g., spelling, semantic, and otherwise) associated with related identifiers are discussed elsewhere.

  enumeration set of named constants identifier learning a list of symbolic name

- *Using a tool to automatically generate possibilities for consideration by the developer*. For instance, Dale and Reiter[82] gave a computational interpretation to the Gricean maxims

- *Asking a large number of subjects to generate possible identifier names,* using the most common suggestions as input to a study of subjects' ability to match and recall the identifiers, the identifier having the best match and recall characteristics being chosen. Such a method has been empirically tested on a small example.[15] However, it is much too time-consuming and costly to be considered as a possible technique in these coding guidelines.

Table 4: Number of identifiers having the same spelling occurring in pairs of programs. *all_prog* denotes the combined source of all the programs. Based on the visible form of the `.c` files.

	all_prog	gcc	idsoftware	linux	netscape	openafs	openMotif	postgresql
all_prog	—	24,271	11,531	182,910	15,411	15,856	12,762	8,196
gcc	24,271	—	1,164	2,896	2,141	1,785	1,241	1,368
idsoftware	11,531	1,164	—	1,895	1,386	1,173	948	847
linux	182,910	2,896	1,895	—	3,058	3,037	1,756	1,719
netscape	15,411	2,141	1,386	3,058	—	2,158	1,818	1,557
openafs	15,856	1,785	1,173	3,037	2,158	—	1,103	1,459
openMotif	12,762	1,241	948	1,756	1,818	1,103	—	905
postgresql	8,196	1,368	847	1,719	1,557	1,459	905	—

2.2.1 Individual biases and predilections

It is commonly believed by developers that the names they select for identifiers are *obvious*, *self-evident*, or *natural*. Studies of people's performance in creating names for objects shows this belief to be false,[50, 110, 111] at least in one sense. When asked to provide names for various kinds of entities, people have been found to select a wide variety of different names, showing that there is nothing *obvious* about the choice of a name. Whether, given a name, people can reliably and accurately deduce the association intended by its creator is not known (if the results of studies of abbreviation performance are anything to go by, the answer is probably not).

[] abbreviating identifier

A good naming study example is the one performed by Furnas, Landauer, Gomez, and Dumais,[110, 111] who described operations (e.g., hypothetical text editing commands, categories in *Swap 'n Sale* classified ads, keywords for recipes) to subjects who were not domain experts and asked them to suggest a name for each operation. The results showed that the name selected by one subject was, on average, different from the name selected by 80% to 90% of the other subjects (one experiment included subjects who were domain experts and the results for those subjects were consistent with this performance). The occurrences of the different names chosen tended to follow an inverse law, with a few words occurring frequently and most only rarely.

[782] zipf's law

Individual biases and predilections are a significant factor in the wide variety of names' selection. Another factor is an individual's past experience; there is no guarantee that the same person would select the same name at some point in the future. The issue of general developer difference is discussed elsewhere. The following subsections discuss some of the factors that can affect developers' identifier processing performance.

[] developer differences

2.2.1.1 Natural language

Developers will have spent significant amounts of time using their native language in both spoken and written forms from an early age. This usage represents a significant amount of learning and many recognition (e.g., recognizing common sequences of characters) and generation (e.g., creating the commonly occurring sounds) operations will have become automatic.

[] automatization

The following natural-language related issues are discussed in the subsequent sections:

- Language conventions, including use of metaphors and category formation.
- Abbreviating of known words.
- Methods for creating new words from existing words.
- Second-language usage.

[] Identifier concepts

[] abbreviating identifier

[] compound word

[] identifier English as second language

[] identifier second language spelling

2.2.1.2 Experience

People differ in the experiences they have had. The following are examples of some of the ways in which personal experiences might affect the choice of identifier spellings.

- *recent experience*. Developers will invariably have read source code containing other identifiers just prior to creating a new identifier. A study by Sloman, Harrison, and Malt[282] investigated how subjects named ambiguous objects immediately after exposure to familiar objects. Subjects were first shown several photographs of two related objects (e.g., chair/stool, plate/bowl, pen/marker). They were then shown a photograph of an object to which either name could apply (image-manipulation software was used to create the picture from photographs of the original objects) and asked to name the object. The results found that subjects tended to use a name consistent with objects previously seen (77% of the time, compared to 50% for random selection; other questions asked as part of the study showed results close to 50% random selection).
- *educational experience*. Although they may have achieved similar educational levels in many subjects, there invariably will be educational differences between developers. A study by Van den Bergh, Vrana, and Eelen[311] showed subjects two-letter pairs (e.g., *OL* and *IG*) and asked them to select the letter pair they liked the best (for "God knows whatever reason"). Subjects saw nine two-letter pairs. Some of the subjects were skilled typists (could touch type blindfolded and typed an average of at least three hours per week) while the others were not. The letter pair choice was based on the fact that a skilled typist would use the same finger to type both letters of one pair, but different fingers to type the letters of the other pair. Each subject scored 1 if they selected a pair typed with the same finger and 0 otherwise. The expected mean total score for random answers was 4.5. Overall, the typists mean was 3.62 and the nontypists mean was 4.62, indicating that typists preferred combinations typed with different fingers. Another part of the study attempted to find out if subjects could deduce the reasons for their choices; subjects could not. The results of a second experiment showed how letter-pair selection changed with degree of typing skill.

naming cultural differences

- *cultural experience*. A study by Malt, Sloman, Gennari, Shi, and Wang[210,211] showed subjects (who were native speakers of either English, Chinese, or Spanish) pictures of objects of various shapes and sizes that might be capable of belonging to either of the categories—bottle, jar, or container. The subjects were asked to name the objects and also to group them by physical qualities. The results found that while speakers of different languages showed substantially different patterns in naming the objects (i.e., a linguistic category), they showed only small differences in their perception of the objects (i.e., a category based on physical attributes).
- *environmental experience*. People sometimes find that a change of environment enables them to think about things in different ways. The environment in which people work seems to affect their thoughts. A study by Godden and Baddeley[121] investigated subjects' recall of memorized words in two different environments. Subjects were divers and learned a list of spoken words either while submerged underwater wearing scuba apparatus or while sitting at a table on dry land. Recall of the words occurred under either of the two environments. The results showed that subjects recall performance was significantly better when performed in the same environment as the word list was learned (e.g., both on land or both underwater). Later studies have obtained environmental affects on recall performance in more mundane situations, although some studies have failed to find any significant effect. A study by Fernández and Alonso[10] obtained differences in recall performance for older subjects when the environments were two different rooms, but not for younger subjects.

2.2.1.3 Egotism

It is not uncommon to encounter people's names used as identifiers (e.g., the developer's girlfriend, or favorite film star). While such unimaginative, ego-driven naming practice may be easy to spot, it is possible that much more insidious egotism is occurring. A study by Nuttin[237] found that a person's name affects their choice of letters in a selection task. Subjects (in 12 different European countries) were given a sheet containing the letters of their alphabet in random order and spaced out over four lines and asked to circle six letters. They were explicitly told not to think about their choices but to make their selection based on those they felt they preferred. The results showed that the average probability of a letter from the subject's name

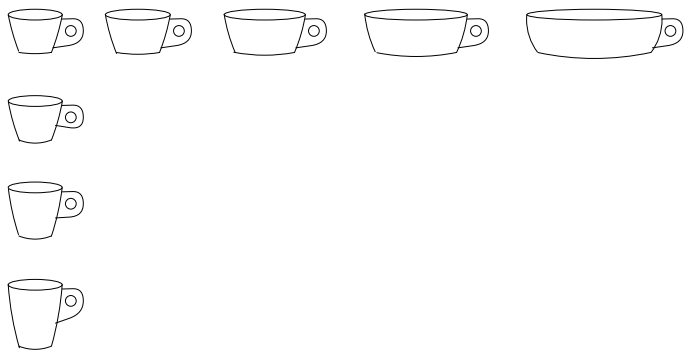


Figure 4: Cup- and bowl-like objects of various widths (ratios 1.2, 1.5, 1.9, and 2.5) and heights (ratios 1.2, 1.5, 1.9, and 2.4). Adapted from Labov.[186]

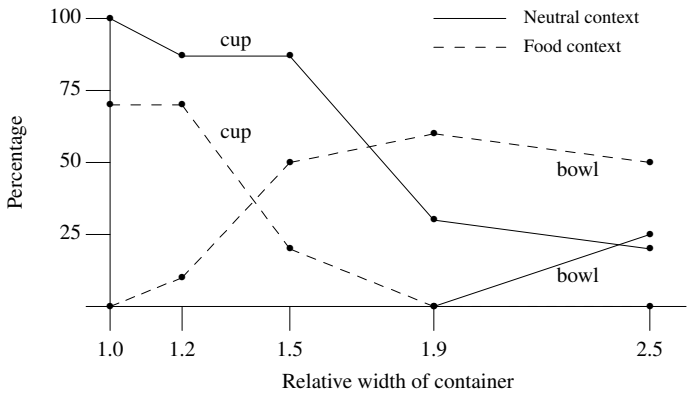


Figure 5: The percentage of subjects who selected the term *cup* or *bowl* to describe the object they were shown (the paper did not explain why the figures do not sum to 100%). Adapted from Labov.[186]

being one of the six chosen was 0.30, while for non-name letters the probability was 0.20 (there was some variation between languages, for instance: Norwegian 0.35 vs. 0.18 and Finnish 0.35 vs. 0.19). There was some variation across the components of each subject's name, with their initials showing greatest variation and greatest probability of being chosen (except in Norwegian). Nuttin proposed that ownership, in this case a person's name, was a sufficient condition to enhance the likelihood of its component letters being more attractive than other letters. Kitayama and Karasawa[174] replicated the results using Japanese subjects.

A study by Jones, Pelham, Mirenberg, and Hetts[164] showed that the amount of exposure to different letters had some effect on subject's choice. More commonly occurring letters were selected more often than the least commonly occurring (a, e, i, n, s, and t vs. j, k, q, w, x, and z). They also showed that the level of a subject's self-esteem and the extent to which they felt threatened by the situation they were in affected the probability of them selecting a letter from their own name.

2.2.2 Application domain context

context naming affected by

The creation of a name for a new identifier, suggesting a semanticly meaningful association with the application domain, can depend on the context in which it occurs.

A study by Labov[186] showed subjects pictures of individual items that could be classified as either cups or bowls (Figure 4). These items were presented in one of two contexts—a neutral context in which the pictures were simply presented and a food context (they were asked to think of the items as being filled with mashed potatoes).

The results show (Figure 5) that as the width of the item seen was increased, an increasing number of

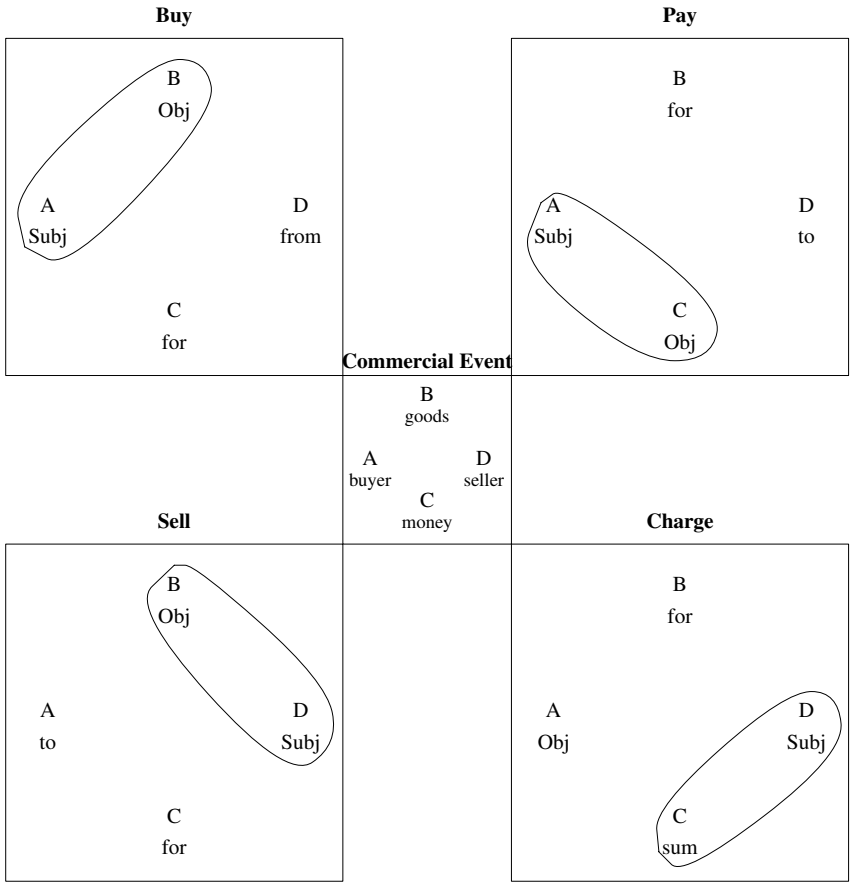


Figure 6: A commercial event involving a buyer, seller, money, and goods; as seen from the buy, sell, pay, or charge perspective. Based on Fillmore.[100]

subjects classified it as a bowl. By introducing a food context subjects responses shifted towards classifying the item as a bowl at narrower widths.

The same situation can often be viewed from a variety of different points of view (the term *frame* is sometimes used); for instance, commercial events include buying, selling, paying, charging, pricing, costing, spending, and so on. Figure 6 shows four ways (i.e., buying, selling, paying, and charging) of looking at the same commercial event.

2.2.3 Source code context

source code context identifier naming conventions identifier other guideline documents hungarian naming identifier identifier encoding usability

It is quiet common for coding guideline documents to recommend that an identifiers spelling include encoded information on the source code context of its declaration. The term *naming conventions* is often used to refer to these recommendations. Probably the most commonly known of these conventions is the Hungarian naming convention,[279] which encodes type information and other attributes in the spelling of an identifier. As discussed elsewhere, such information may not be relevant to the reader, may reduce the memorability of the identifier spelling, may increase the probability that it will be confused with other identifiers, and increase the cost of maintaining code.

The two language contexts that are used to influence the spelling of identifiers are namespace and scope. The following subsections briefly discusses some of the issues and existing practices.

2.2.3.1 Name space

Macro naming conventions

macro naming conventions

There is a very commonly used convention of spelling macro names using only uppercase letters (plus underscores and digits; see Table 5). Surprisingly this usage does not consume a large percentage of available character combinations (3.4% of all possible four-character identifiers, and a decreasing percentage for identifiers containing greater numbers of characters).

The use of uppercase letters for macro names has become a C idiom. As such, experienced developers are likely to be practiced at recognizing this usage in existing code. It is possible that an occurrence of an identifier containing all uppercase letters that is not a macro name may create an incorrect belief in the mind of readers of the source.

There are no common naming conventions based on an identifier being used as a macro parameter. The logical line based nature of macro definitions may result in macro parameter names containing only a few characters having less cost associated with them than those containing many characters.

macro parameter naming conventions

[] preprocessor directives syntax

Tag and typedef naming conventions

tag naming conventions

There is a commonly seen naming convention of giving a tag name and an associated typedef name the same spelling (during the translation of individual translation units of the books benchmark programs 30% of the tag names declared had the same spelling as that used in the declaration of a typedef name). Sharing the same name has advantage of reducing the amount of information that developers need to remember (once they have learned this convention). However, apart from existing practice, there is no obvious benefit in having two different methods of denoting the same type.

Given that one of three keywords immediately precedes a tag name, its status as a tag is immediately obvious to a reader of the source (the only time when this context may not be available is when a tag name occurs as an argument in a macro invocation). Given the immediate availability of this information there is no benefit in a naming convention intended to flag the status of an identifier as a tag.

[] syntactic context

The following are several naming conventions that are often seen for typedef names. These include:

typedef naming conventions

[782] typedef name no lowercase

- No lowercase letters are used (i.e., uppercase letters, digits, and underscore are used).
- Information on the representation of the type is encoded in the spelling. This encoding can vary from the relatively simply (e.g., `INT_8` indicates that an object is intended to hold values representable in an integer type represented in eight bits; a convention that is consistent with that used in the **`<stdint.h>`** header), or quiet complex (e.g., hungarian naming).

[] MISRA

[] stdint.h header

[782] hungarian naming identifier

It is possible for type information, in an identifiers spelling, to be either a benefit or a cost, for readers of the source. For instance, readers may assume that the following equality holds `sizeof(INT_8) == sizeof(char)`, when in fact the author used type **`int`** in the declaration of all `INT_` typedef names.

Member naming conventions

member naming conventions

Some coding guideline documents recommend that the names of members contain a suffix or prefix that denotes their status as members. The cost/benefit of specifying this information in the spelling of an identifier name is discussed elsewhere.

[] member namespace

Label naming conventions

label naming conventions

There are no common C naming conventions for identifiers that denote labels. However, some coding guideline documents recommend that label names visually draw attention to themselves (e.g., by containing lots of characters). Label name visibility was an explicit goal in the specification of the syntax of labels in Ada. Other coding guideline documents recommend that label names not be visible at all (i.e., they only appear within macro replacement lists).

[] labeled statements syntax

Given that identifiers denoting label names can only occur in two contexts, and no other kinds of identifiers can occur in these contexts, there is no benefit in encoding this information (i.e., is a label) in the spelling. Whether it there is a worthwhile cost/benefit in visually highlighting the use of a label needs to be evaluated on a usage by usage basis. There are a variety of techniques that can be used to provide visual highlighting, it is not necessary to involve an identifiers spelling.

enumeration constant naming conventions

Enumeration constant naming conventions

Some coding guideline documents recommend that the names of members contain a suffix or prefix (e.g., `E_` or `_E`) that denotes their status as members. Unlike member and label names it is not possible to deduce that an identifier is an enumeration constant from the syntactic context in which it occurs. However, there does not appear to be a worthwhile cost/benefit in encoding the status of an identifier as an enumeration constant in its spelling.

The issue of selecting the names of enumeration constants defined in one enumeration type to form a distinct set of symbols is discussed elsewhere.

enumeration set of named constants

function naming conventions

Function naming conventions

Some coding guideline documents recommend that the names of functions contain a verb (sometimes a following noun is also specified). A study by Caprile and Tonella[48] created a word grammar describing function names (which was structured in terms of actions) and were able to parse a large percentage of such names in a variety of programs (80% in the case of the mosaic sources).

scope naming conventions

2.2.3.2 Scope

Tools that automatically generate source code might chose to base part of the spelling of an identifier on its scope to simplify the task of writing the generator. If names followed a fixed unusual, pattern the possibility of duplicates being declared is likely to be reduced.

file scope naming conventions

identifier other guideline documents

File scope

Some coding guideline documents require identifiers declared in file scope to include a prefix denoting this fact (it is rare to find suffixes being used). The reasons given for this requirement sometimes include issues other than developer readability and memorability; one is management control of globally visible identifiers (exactly why management might be interested in controlling globally visible identifiers is not always clear, but their authority to silence doubters often is).

What are the attributes of an identifier at file scope that might be a consideration in the choice of its name?

- They are likely to be referenced from many function definitions, (unlike block scope identifiers a readers knowledge of them needs to be retained for longer periods of time).
- They are unlikely to be immediately visible while a developer is looking at source code that references them (unlike block scope identifiers, their declaration is likely to be many lines—hundreds—away from the points of reference).
- They will be unique (unlike block scope names, which can be reused in different function definitions).

During code maintenance new identifiers are often defined at file scope. Does the choice of spelling of these file scope identifiers need to take account of the spelling of all block scope identifiers defined in source files that **`#include`** the header containing the new file scope declaration? The options have the following different costs:

1. Changing the spelling of any block scope identifiers, and references to them, to some other spelling. (This will be necessary if the new, file scope identifier has identical spelling and access to it is required from within the scope in which the local identifier is visible.) There is also the potential cost associated with the block scope identifier not having the ideal attributes, plus the cost of developer relearning associated with the change of an existing identifier spelling.

identifier primary spelling issues

2. Selecting another spelling for the file scope identifier. To know that a selected spelling clashes with another identifier requires that the creator of the new identifier have access to all of the source that **`#include`** the header containing its declaration. There is also the potential cost associated with the file scope identifier not having the ideal attributes. There is no relearning cost because it is a new identifier.

identifier primary spelling issues

3. Accepting the potential cost of deviating from the guideline recommendation dealing with identifier spellings.

Each of these options has different potential benefits; they are, respectively:

1. The benefits of following the identifier spelling guideline recommendations are discussed elsewhere. The benefit is deferred. [] identifier primary spelling issues
2. No changes to existing source need to be made, and it is not necessary for developers declaring new file scope identifiers to have access to all of the source that **`#include`** the header containing its declaration. The benefit is deferred.
3. There is no benefit or immediate cost. There may be a cost to pay later for the guideline deviation.

Block scope block scope naming conventions

Because of their temporary nature and their limited visibility some coding guideline documents recommend the use of short identifiers (measured in number of characters) for block scope object definitions. What is the rationale for this common recommendation?

Some developers openly admit to using short identifiers because they are quicker to type. As pointed out elsewhere, the time taken by a developer to type the characters of an identifier is not significant, compared to the costs to subsequent readers of the source code of a poorly chosen name. Your author suspects that it is the cognitive effort required to create a meaningful name that many developers are really trying to avoid. [] typing minimization

What are the properties of identifiers, in block scope, that might be a consideration in the choice of their names?

- They are likely to appear more frequently within the block that defines them than names having file scope (Figure).
- The semantic concepts they denote are likely to occur in other function definitions.
- A program is likely to contain a large number of different block scopes.
- Their length is likely to have greater impact on the layout of the source code than other identifiers.
- Translators do not enforce any uniqueness requirements for names appearing in different block scopes.
- They need to be memorable only while reading the function definition that contains them. Any memories remaining after that block has been read should not cause confusion with names in other function definitions.

2.2.4 Suggestions for spelling usage

identifier suggestions

The following list provide suggestions on how to make the best use of available resources (a reader's mental capabilities) when creating identifier spellings. The studies on which these suggestions are based have mostly used English speakers as subjects. The extent to which they are applicable to developers readers of non-English languages is not known (other suggestions may also be applicable for other languages). [782] identifiers Greek readers

These suggestions are underpinned by the characteristics of both the written and spoken forms of English and the characteristics of the device used to process character sequences (the human brain). There is likely to be a great deal of interdependence between these two factors. The characteristics of English will have been shaped by the characteristics of the device used to create and process it.

- *Delimiting subcomponents.* Written English separates words with white space. When an identifier spelling is composed of several distinct subcomponents, and it is considered worthwhile to provide a visual aid highlighting them, use of an underscore character between the subcomponents is the closest available approximation to a reader's experience with prose. Some developers capitalize the first letter of each subcomponent. Such usage creates character sequences whose visual appearance are unlike [] words white space between

those that readers have been trained on. For this reason additional effort will be needed to process them. In some cases the use of one or more additional characters may increase the effort needed to comprehend constructs containing the identifier (perhaps because of line breaks needed to organize the visible source). Like all identifier spelling decisions a cost/benefit analysis needs to be carried out.

initial letters
identifier
identifier recall
words
English

- *Initial letters.* The start of English words are more significant than the other parts for a number of reasons. The mental lexicon appears to store words by their beginnings and spoken English appears to be optimized for recognizing words from their beginnings. This suggests that it is better to have differences in identifier spelling at the beginning (e.g., *cat*, *bat*, *mat*, and *rat*) than at the end (e.g., *cat*, *cab*, *can*, and *cad*).

- *Pronounceability.* Pronounceability may appear to be an odd choice for a language that is primarily read, not spoken. However, pronounceability is an easy-to-apply method of gauging the extent to which a spelling matches the characteristics of character sequences found in a developers native language. Given a choice, character sequences that are easy to pronounce are preferred to those that are difficult to pronounce.

memory
chunking

- *Chunking.* People find it easier to remember a sequence of short (three or four letters or digits) character sequences than one long character sequence. If a non-wordlike character sequence has to be used, breaking the character sequence into smaller chunks by inserting an underscore character between them may be of benefit to readers.

- *Semantic associations.* The benefits of identifier spellings that evoke semantic associations, for readers are pointed out in these and other coding guideline documents. However, reliably evoking the desired semantic associations in different readers is very difficult to achieve. Given a choice, an identifier spelling that evokes, in many people, semantic associations related to what the identifier denotes shall be preferred to spellings that evoke them in fewer people or commonly evokes semantic associations unrelated to what the identifier denotes.

- *Word frequency.* High-frequency words are processed more rapidly and accurately than low-frequency words. Given a choice, higher-frequency words are be preferred to lower-frequency words.

word frequency 782

2.2.4.1 Existing conventions

In many cases developers are adding identifiers to an existing code base that already contains thousands, if not tens of thousands, of identifiers. The maintainers of this existing code will have learned the conventions used (if only implicitly). Having new identifier spellings follow existing conventions enables maintainers to continue to obtain the benefits from what they have learned, and does not increase costs by requiring that exceptions be handled or new conventions learned. Following an existing convention has a benefit in its own right, independently of the original reasons (which may not even have been valid at the time) for adopting it.

implicit learning

One problem with existing source code conventions is finding out what they are. It is possible that the conventions used will vary across the files making up a program (perhaps following the habits and characteristics of the original authors). The following discussion attempts to highlight the main *convention domains* that affect identifier spellings:

- Natural language usage conventions. For instance, word order and creating new words by joining together—*compounding*—existing words. There are often rules for ordering and compounding words and speakers of languages often are sensitive to these rules (usually implicitly following them without consciously being aware of it or even explicit having knowledge of what the rules are).

implicit learning

- General software development conventions. For instance, using the abbreviation `ptr` to denote something related to pointers.

- C language developer conventions. For instance, using uppercase letters for identifier spellings denoting macro definitions or typedef names.

[782] macro naming conventions
[782] typedef naming conventions

- Development group conventions. It is your author's experience that these are rarely effectively enforced and noncompliance is common (even among developers attempting to follow them).[6] Without measurements confirming that any development group guidelines are followed in the source being maintained, it is suggested that these conventions be ignored from the point of view of achieving a benefit by considering them when creating new identifier spellings. However, claiming to follow them may reduce the effort of dealing with management, a topic that is outside the scope of this book.
- Host environment conventions. Developers who primarily work within a particular host environment (e.g., Linux or Microsoft Windows) often follow conventions specific to that environment. Whether this is because of the influence of a particular vendor or simply the drifting apart of two communities, is not known.
- An individual's past experience. For all the supposed dynamism of the software business, developers can be remarkably conservative in their identifier-naming habits, often being very resistant to change.

Table 5: Occurrence of declared identifiers (as a percentage of all identifiers in the visible form of the `.c` files; weighted by number of occurrences; unique identifiers are in parentheses) containing particular character sequences (the phrase *spelled using upper-case letters* is usually taken to mean that no lower-case letters are used, i.e., digits and underscore are included in the possible set of characters; for simplicity and accuracy the set of characters omitted are listed).

	no lower-case	no upper-case	no underscore	no digits	only first character upper-case
file scope objects	0.8 (1.0)	80.3 (79.1)	29.6 (25.4)	87.3 (85.7)	5.2 (5.7)
block scope objects	1.3 (1.8)	91.9 (81.3)	79.9 (58.9)	96.3 (93.0)	1.3 (3.1)
function parameters	0.1 (0.4)	94.2 (82.9)	88.6 (67.4)	96.8 (94.8)	1.4 (2.9)
function definitions	0.2 (0.2)	59.0 (62.1)	27.1 (24.1)	87.1 (86.4)	29.9 (27.3)
struct/union members	0.5 (0.8)	78.5 (71.8)	65.7 (51.3)	93.2 (91.4)	12.0 (14.2)
function declarations	0.7 (0.5)	55.5 (57.1)	27.3 (26.5)	88.7 (87.5)	32.4 (30.1)
tag names	5.7 (6.6)	60.7 (63.8)	25.6 (21.6)	88.1 (85.9)	18.4 (14.5)
typedef names	14.0 (17.0)	37.0 (33.5)	45.0 (40.4)	89.7 (89.3)	39.8 (37.4)
enumeration constants	55.8 (56.0)	10.8 (10.6)	16.0 (15.0)	79.9 (77.9)	32.1 (32.0)
label names	27.2 (48.1)	69.2 (47.4)	70.8 (65.6)	67.4 (46.3)	2.2 (2.3)
macro definitions	78.4 (79.9)	4.9 (5.0)	15.5 (13.0)	70.9 (69.3)	13.1 (11.1)
macro parameters	19.8 (20.4)	77.6 (68.7)	96.0 (83.6)	94.2 (90.7)	1.4 (5.0)

2.2.4.2 Other coding guideline documents

identifier other guideline documents

Other coding guideline documents invariably include recommendations on identifier spelling. These recommendations are sometimes vague to the point of near worthlessness (e.g., "Use meaningful names for identifiers"), or the over-exaggeration of the importance of certain kinds of information with an associated lack of consideration of all the factors involved with identifier usage. Examples of the latter include:

- *Hungarian notation.* The information provided by this notation is often not needed by the reader, who unnecessarily has to process characters that are essentially noise. This notation also creates maintenance effort in that identifier spellings have to be updated when their types are changed.

[782] hungarian naming identifier

- *Meaningful names as brief sentences.* Here priority is given to semantic associations created by an identifiers spelling. The disadvantages of using large numbers of characters in an identifier spelling is discussed elsewhere.

[] identifier number of characters

[6] This experience comes from many onsite visits where a development group's source code was analyzed by a tool configured to enforce that group's identifier-naming conventions.

- *Short names for local identifiers.* Here priority is given to the effort needed to type identifiers and potential use of short-term memory resources (shorter names are likely to require less time to pronounce).
- *Use of prefixes.* Adding prefixes to words has a long history. French scribes in the middle ages would add an *h* to the start of words that were derived from Latin words that started with the letter *h*.[220] The *h* not being pronounced (e.g., modern French *habile* and *honneur*). The introduction of these words into English resulted in either the *h*'s being dropped (*able*), remaining silent (*honour*), or causing a change of pronunciation (*hospital*). The importance of the initial letters, at least for native English speakers, is pointed out above. Mandating the use of prefixes is equivalent to specifying that the information they denote is more important than any other information evoked by an identifiers spelling. If this is true the recommendation to use prefixes is correct, otherwise it causes needless waste of a readers cognitive resources.
- *No lowercase letters are used (that is, uppercase letters, digits, and underscore are used) for macro definitions and typedef names.*—This usage appears to give priority to signaling implementation details to the reader. (While there is a prose text convention, at least in English, that words written using all uppercase letters denote important text, this usage in source code is more of a visual convention than an indication that these kinds of identifiers are more important than others.) Using uppercase letters for macro definitions prevents them from being treated as interchangeable with function definitions (at least for function-like macros). Requiring that macro names be spelled using only uppercase letters creates additional work if the source code is modified. For instance, a macro name that is changed to an enumerated constant or is replaced by an object either has to remain in uppercase or be converted to lowercase. Similarly for a function definition that is remapped to a macro definition (going from lowercase to uppercase). Typedef names appear less often in source code than other kinds of ordinary identifiers. While the syntactic context in which they appear signifies the kind of identifier they represent, readers are likely to be expecting to see a keyword in these contexts (this is the common case). When readers are quickly scanning the source, use of all uppercase letters in the spelling of a typedef name may provide an alternative visual mechanism for rapid recognition (potentially reducing the effort needed to rapidly scan source).

typedef name no lowercase

- *Management/project control.* Here priority is given to information used in the management and coordination of a programs source code. Reserving a set of identifier spellings, usually for future usage, sometimes occurs.

reserved identifier

Some coding guideline documents apply identifier-naming conventions that might be applicable in some programming languages (e.g., Cobol[236]) but are not applicable in C (in the case of Cobol because of the different declaration syntax and in some cases semantics).

identifier filtering spellings

2.3 Filtering identifier spelling choices

This subsection makes guideline recommendations on what identifier spellings should not be used. It does not aim to extensively discuss other spelling filtering issues that developers might consider, although some are covered. It is unlikely to be practical for developers to manually apply these guideline recommendations. Automatic enforcement is assumed to be the most likely method of checking adherence to these recommendations. Whether this automated process occurs at the time an identifier is declared, or sometime later, is a practical cost/benefit issue that is left to the developer to calculate.

optimal spelling 782 identifier

The discussion on creating optimal identifier spellings pointed out the need to consider all identifiers declared in the translation of a program. However, the computational cost of considering all identifiers is significant and the guideline recommendations that follow often specify a smaller set of possible identifier spellings that need to be considered.

The basis for these filtering recommendations is the result of the studies described in the major subsections following this one. The major issues are the characteristics of the human mind, the available cognitive resources (which includes a reader's culture and training), and usability factors.

The basic assumption behind the guideline recommendations is that a reduction in similarity between identifiers will result in a reduction in the probability that readers will mistake one for another. The similarity between two identifiers is measured using the typed letters they contain, their visual appearance, and spoken and semantic forms.

2.3.1 Cognitive resources

Other subsections of this coding guideline separate out discussion of issues relating to the functioning of the human brain, and cultural and educational factors. Here they are grouped together as cognitive resources.

An algorithm for calculating the cognitive resources needed to process an identifier spelling is not yet available. For simplicity the following discussion treats each resource as being independent of the others.

2.3.1.1 Memory factors

The primary human memory factors relevant to the filtering of identifier spellings are the limited capacity of short-term memory and its sound-based operating characteristics. The STM capacity limitation issues associated with identifier spelling are discussed elsewhere.

[] memory developer
[] identifier STM required

If two identifiers are both referenced in related sections of source code, it is possible that both of their pronunciations will be held in a reader's phonological loop (audio short-term memory) at the same time. The following guideline recommendation is intended to reduce the likelihood of interference, in short-term memory, between two similar sounding identifier spellings.

[] phonological loop

REV A newly declared identifier shall have a Levenstein distance, based on the phonemes in its spoken form, of at least two from existing identifiers declared in a program.

This guideline recommendation may be overly restrictive, preventing otherwise acceptable spellings from being used. The following deviation is based on studies showing, at least for native-English speakers, that the start of a word has greater salience than its middle or end.

DEV A newly declared identifier may have a Levenstein distance, based on phonemes, of one from existing identifiers declared in a program provided the difference occurs in the first phoneme.

2.3.1.2 Character sequences

Every natural language has patterns to the way it joins sounds together to form words. This in turn leads to patterns (at least for nonlogographic orthographies) in the character sequences seen in the written form (even in an irregularly spelled language such as English). Proficient users of a language have overlearned these patterns and can effortlessly recognize them.

[782] logographic
[782] orthography

While novice readers may read words a letter at a time, experts make use of their knowledge of commonly occurring character sequences (which may be complete words) to increase their reading rate. The penalty for making use of statistical information is an increased likelihood of making mistakes, particularly when reading character sequences that are not required to be words (e.g., identifier spellings).

```
1   enum {OO, 00, l1, ll} glob;
```

Word recognition is driven by both bottom-up processes (the visible characters) and top-down processes (reader expectations). Minimizing the visual similarity between identifier spellings is one technique for reducing the likelihood of a reader mistakenly treating one identifier for another, different, identifier. Although the data needed to calculate an accurate value for the visual similarity between two identifiers is not yet available, the following guideline recommendation is still considered to be worth making.

[] identifier visual similarity

[3] A newly declared identifier shall have a Levenstein distance, based on visual similarity of corresponding characters, of at least two when compared against all identifiers declared in the visible source of a program.

For the purpose of this guideline recommendation, the visual similarity Levenstein of two identifiers is defined as the sum, over all pairs of characters, of the visual distance between two characters (one from each identifier) occurring at the same position in the identifier spelling (a space character is used to pad the shorter identifier). The visual distance between two characters is defined as (until a more accurate metric becomes available):

1. zero if they are the same character,
2. zero if one character represents the letter *O* (uppercase oh) and the other is the digit zero,
3. zero if one character represents the letter *l* (lowercase ell) and the other is the digit one,
4. otherwise, one.

identifier semantic associations

name space

2.3.1.3 Semantic associations

The spelling of an identifier is assumed to play a significant role in a readers recall of the semantic information associated with it (another factor is the context in which the identifier occurs). Having two different identifiers with the same spelling and:

- with different semantic information associated with them is likely to be create a cost (i.e., recall of information associated with other identifiers sharing the same spelling),
- with the same semantic information associated with them is likely to be create a benefit (i.e., improved recall performance).

power law of learning

[4] A newly declared identifier shall not have the same spelling as another identifier declared in the same program.

DEV A newly declared identifier may have the same spelling as another identifier declared in the same program provided they are both used to denote the same information and both have block scope.

Some identifiers are formed by concatenating two or more known words, abbreviations, or acronyms (these subcomponents are called *conceptual units* here). The interpretation given to a sequence of these conceptual units may not depend significantly on their relative ordering. For instance, either `widget_total` or `total_widget` might be considered to denote a count of the total number of widgets.

The following guideline recommendation is motivated by possible semantic confusion, not by the possibility of typing mistakes. While the general block-edit string matching problem is NP-complete,[201] limiting the comparison to known *conceptual units* significantly reduces the computational cost of checking adherence.

REV A newly declared identifier shall have a Levenstein distance, based on individual conceptual units, of at least two from existing identifiers declared in a program.

DEV A newly declared identifier defined in a function definition shall have a Levenstein distance, based on individual conceptual units, of at least two from existing identifiers defined in other function definitions.

In some cases developers may consider two identifiers, differing by a Levenstein distance of one, to be semantically distinct. For instance, `widget_num` or `num_widget` might be considered to denote a number assigned to a particular widget and some count of widgets, respectively. Such an interpretation is dependent on knowledge of English word order and conventions for abbreviating sentences (e.g., "widget number 27" and "number of widgets"). However, this distinction is much too subtle and relies on too fine a point of interpretation (and could quite easily be given the opposite interpretation) for any form of deviation to be justified.

2.3.2 Usability

identifier encoding usability

The following discussion briefly summarizes the general issues associated with identifier. This issue is discussed in more detail elsewhere.

[] identifier usability

2.3.2.1 Typing

Developers make mistakes when typing the characters that form an identifier spelling. If two identifier spellings differ by a single character, it is possible that an uncorrected mistake will cause a different identifier to be accessed.

[6] A new identifier shall have a Levenstein distance, based on individual characters, of at least two from existing identifiers declared in a program.

An identifier may differ by a Levenstein distance of one from another identifier and not be accessible at the point in the source a typing mistake occurs because it is not visible at that point. Requiring a Levenstein distance of two for all new identifiers may be overly restrictive (preventing otherwise acceptable spellings from being used).

[DEV] An identifier defined in a function definition may have a Levenstein distance, based on individual characters, of one from existing identifiers defined in other function definitions.

2.3.2.2 Number of characters

identifier number of characters

The C Standard minimum requirements on the number of significant characters in an identifier spelling (the first 31 in an external identifier, and 63 in an internal linkage or macro name) is not usually an issue in human-written code. The issue of translators that have yet to support the new limits (the requirements specified in the previous version of the Standard were lower) are discussed in the limits sentences.

[] external identifier significant characters

[] internal identifier significant characters

2.3.2.3 Words unfamiliar to non-native speakers

The usual reason for including a word in an identifier spelling is to obtain the benefit of the semantic associations it evokes. If the source is likely to be maintained by developers whose native language is different from that of the author, it is necessary to consider the possibility that some character sequences will not be recognized as words.

Experience shows that technical words often cause the fewest problems. Within a particular application domain, the vocabulary of technical words used at the source code level is usually relatively small (compared to natural languages and even the specific application domain). They are relatively easy for native speakers to identify and L2 speakers may not be embarrassed by their ignorance of these technical terms.

Identifying nontechnical words that may be unfamiliar to non-native speakers is often more difficult. Native speakers rarely have the necessary experience and asking non-native speakers about their language competence may be awkward or impractical.

Although there have been some surveys of L2 vocabulary knowledge,[159] the available information does not appear to be sufficiently extensive to enable a guideline recommendation (that words likely to be unfamiliar to L2 speakers not be used) to be enforced; so none is given here.

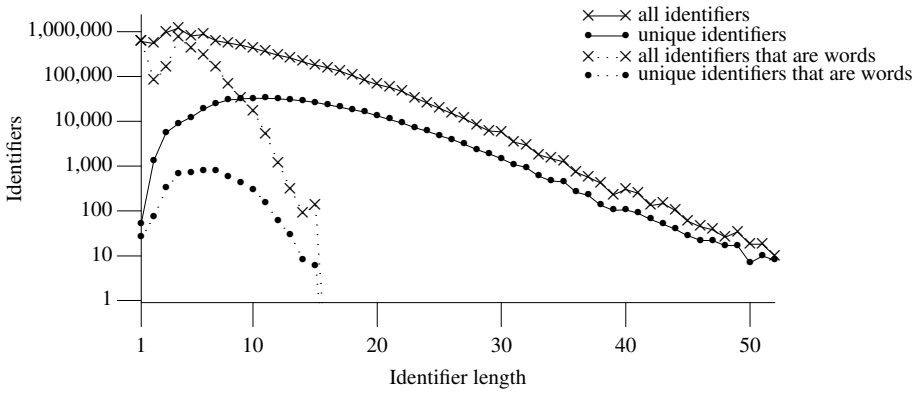


Figure 7: Number of identifiers (unique and all) of different length in the visible form of the `.c` files. Any identifier whose spelling appeared in the `aspell` 65,000 word dictionary was considered to be a word.

2.3.2.4 Another definition of usability

In some cases identifier spelling usability might be defined in terms of satisfying a management requirement other than minimizing future maintenance costs. For instance, the customer may require adherence to a coding guideline document, which recommends that various kinds of semantic information be explicitly encoded in an identifier's spelling (e.g., the MISRA C Guidelines contain an advisory rule that the spellings of typedef names contain information on the number bits in the storage representation of the defined arithmetic type).

MISRA

Definitions of usability based on these alternative requirements are not discussed further in this coding guideline section.

human language characteristics

3 Human language

This section discusses the characteristics of human languages, primarily in their written form. These empirical and theoretical findings provide background material for the discussion on the three primary requirement issues (memorability, confusability, usability). The section concludes with a more detailed discussion of one language, English.

3.1 Writing systems

orthography

A writing system, known as an *orthography*, uses written characters to represent the structure of a linguistic system. If the character-to-sound rules are relatively simple and consistent (e.g., German, Spanish, Japanese hirigana), the orthography is said to be *shallow*; while if they are complex and inconsistent (e.g., English), it is said to be *deep*. A writing system is more than a set of characters and the sequences of them used to represent words; there are also the conventions adopted by its writers (e.g., direction of writing and some written abbreviations have no equivalent spoken form).

Table 6: Number of people using particular types of writing system for the top 50 world languages in terms of number of speakers. Literacy rates from UNESCO based on typical countries for each language (e.g., China, Egypt, India, Spain). Adapted from Cook.[69]

Total languages out of 50	Speakers (millions)	Readers (millions, based on illiteracy rates)
Character-based systems—8 (all Chinese) + Japanese	1,088	930
Syllabic systems—13 (mostly in India) + Japanese, Korean	561	329
Consonantal systems—4 (two Arabic) + Urdu, Persian	148	no figures available
Alphabetic systems—21 (worldwide)	1,572	1,232

Most existing source code is written using only characters from the basic source character set. The introduction of Universal Character Names and growing tool support for extended characters continues to increase the likelihood that developers will encounter identifiers spelled using characters outside of the invariant Latin subset.

[] basic source character set
[] universal character name syntax
[] ISO 646

There are three kinds of writing systems:

1. *alphabetic.* These writing systems contain a small set of letters. Sequences of one or more of these letters represent the basic spoken units of a word (this sequence of letters is known as a *grapheme* and the sound units it represents is a phoneme) or a complete word. One or more graphemes may be written in sequence to represent a spoken word. This writing system has two varieties:

grapheme
[782] phoneme

   - Abjads, or consonant alphabets, predominantly use only consonants in their written forms. Vowels can be added, usually by means of diacritics, but this is not common (Arabic and Hebrew use them in poetry and children's books). Most abjads, with the exception of Divehi hakura and Ugaritic, are written from right-to-left.
   - Alphabets, or phonemic alphabets, nearly always represent consonants and vowels in written works (acronyms may not contain any vowels).

   Some scripts, for instance Arabic, are used both as an abjad and as an alphabet.

2. *syllabic.* These writing systems use individual characters to represent syllables; for instance, Bengali, Cherokee, and Japanese Katakana.

syllabic
[782] syllable

3. *logographic.* These writing systems, or logosyllabaries, are the most complex natural language writing systems. They can be broken down into the following:

logographic

   - Logograms are symbols that represent whole words, without a phonetic component. Some logograms resemble, or originally resembled, the things they represent and are sometimes known as *pictograms* or *pictographs*.
   - Ideograms are symbols that graphically represent abstract ideas, without a phonetic component.
   - Semantic–phonetic compounds are symbols that include a semantic element, which represents or hints at the meaning of the symbol, and a phonetic element, which denotes or hints at the pronunciation. Some of these compound symbols can be used for their phonetic value alone, without regard for their meaning.

   Examples include Chinese, Japanese Kana, and Ancient Egyptian.

3.1.1 Sequences of familiar characters

When a writing system breaks the representation of a word into smaller components (e.g., alphabetic, phonemic, or syllabic), readers learn not only the visual form of the characters, but also implicitly learn the likelihood of encountering particular character sequences. For instance, readers of English would expect the letter *t* to be followed by *h* and would be surprised to see it followed by *q*. Information on frequency of occurrence of letter sequences in different languages is sufficiently reliable that it is used by cryptographers to help break codes,[114] by OCR software for correcting errors in scanned text, and by mobile phones for predictive text entry.

The frequency of occurrence of particular letter sequences varies between different languages. It will depend on how letter sequences are mapped to phonemes and the frequency of the different phonemes used in the spoken form of a language.

phoneme 782

In his famous paper, *A Mathematical Theory of Communication* Shannon[276] gave the following example of letter sequences that successively approximated English.

- Zero-order approximation (symbols independent and equiprobable): XFOML RXKHRJFFJUJ ZLPWCFWKCYJ FFJEYVKCQSGHYD QPAAMKBZAACIBZLHJQD.
- First-order approximation (symbols independent but with frequencies of English text): OCRO HLI RGWR NMIELWIS EU LL NBNESEBYA TH EEI ALHENHTTPA OOBTTVA NAH BRL.
- Second-order approximation (digram structure of English): ON IE ANTSOUTINYS ARE T INCTORE ST BE S DEAMY ACHIN D ILONASIVE TUCOOWE AT TEASONARE FUSO TIZIN ANDY TOBE SEACE CTISBE.
- Third-order approximation (trigram structure of English): IN NO IST LAT WHEY CRATICT FROURE BIRS GROCID PONDENOME OF DEMONSTURES OF THE REPTAGIN IS REGOACTIONA OF CRE.

The entropy of English has been estimated as 1.75 bits per character[302] (a random selection from 26 letters or space would have an entropy of 4.75 bits per character). It is this predictability, coupled with a reader's knowledge of it, built up through years of practice, that enables people to process words from their first language much more quickly than words from an unfamiliar language. However, a reader's rate of information extraction does not increase; they simply learn to take account of the redundancy present in the input.

identifier information extraction

Information on common letter sequences has been used in models of reader eye movements and typing performance.

Mr. Chips
typing mistakes

Many of the top 12 languages (Table 3) use letters from the invariant Latin character set (with the exception of English, they also include a few additional characters). Because these languages each use slightly different sets of phonemes and use different letters sequences to represent them, letter sequences that are common in one language may be rare in another.

language
Latin characters

If an identifier spelling contains a word belonging to some natural language, readers unfamiliar with that language may break the character sequence up into different letter sequences than the original author did. The letter sequences may be shorter, possibly a single letter. Some of the issues involved in a reader's handling of unknown character sequences, nonwords, is discussed elsewhere.

word pronounceability
nonword effects 782
characters mapping to sound
identifier nonword spelling
reading characters unknown to reader

3.1.2 Sequences of unfamiliar characters

Developers do not need to know anything about any human language, expressed using a particular writing system, to comprehend source code containing instances of that writing system. To comprehend source code readers need to be able to

- deduce when two identifiers have the same, or different, spellings. This same/different task only requires the ability to visually compare two identifiers.

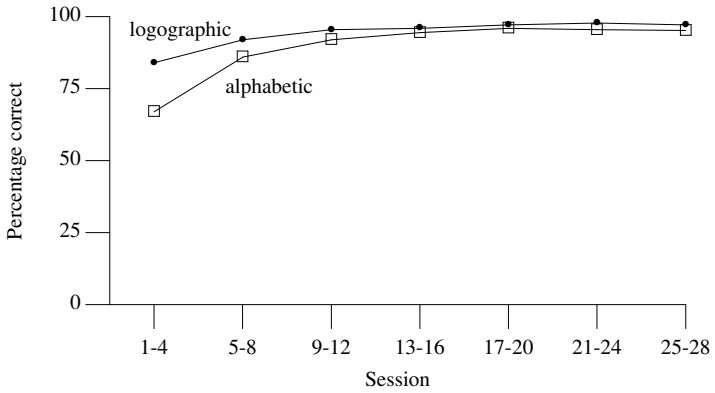


Figure 8: Improvement in word-recognition performance with number of sessions (most sessions consisted of 16 blocks of 16 trials). Adapted from Muter and Johns.[229]

- store a representation of some previously seen identifiers (those considered worth remembering). This task requires a mapping from the visual form of the identifier to some internal, in the developer's mind, representation.

How do readers perform when presented with an unfamiliar writing system (e.g., written using universal character names)?

[] universal character name syntax

A study by Brooks[38] investigated subject's performance in identifying character sequences built from characters they were unfamiliar with—for instance, Π*)(*Π*()*. One group of subjects was first taught a character-to-letter mapping (Π⇒*N*, *)(*⇒*E*, Π⇒*A*, *()*⇒*P*), while the other group was not. Both groups of subjects were asked to learn associations between a number of character sequences and a corresponding spoken response. The spoken form of each character sequence corresponded to the English pronunciation of the word formed by mapping the characters to the letters (which only one group of subjects had been taught). Subject's performance (time taken to speak all character sequences, presented to them in a random order, they had learned) was measured. The results showed an initial performance advantage for subjects using purely visual recognition (they had not been taught a mapping to letters). The performance of both groups of subjects improved with practice. However, given sufficient practice, the performance of subjects who had learned the character to letter mapping exceeded that of subjects who had not learned it. The amount of practice needed seemed to depend on the number of character sequences that had to be learned and their visual form.

A study by Muter and Johns[229] asked subjects (native speakers of English) to learn to identify either a set of logographs (Chinese characters) or words written in an unfamiliar alphabetic code (Devanagari—the written form of Hindi). As expected subjects reaction time and error rates improved with practice. However, the initial performance with logographs was significantly better than alphabetic codes (Figure 8).

The result of this and other character learning studies shows that people relatively quickly achieve high proficiency. However, previous experience with an alphabetic character set would not seem to confer any advantage during the initial learning phrase of a new set of alphabetic characters.

[] letter detection

3.2 Sound system

word sound system

The sounds used by individual natural languages do not use all possible combinations that the human vocal tract is capable of producing.[25] While there may be thousands of possibilities, a particular language usually uses less than a hundred (English uses approximately 44, depending on regional accent).

phoneme

The *phoneme* is the minimal segment of sound that distinguishes one word from another word. These are generally divided into vowels (open sounds, where there are no obstructions to the flow of air from the mouth; known languages contain from 2 to 25 vowels) and consonants (created by some form of obstruction

to the passage of air through the speech tract; known languages contain between 5 and more than 100 consonants). Hawaiian contains 12 phonemes, while !Xu (spoken near the Kalahari desert in Southern Africa) has as many as 141. The most commonly encountered vowels, in different languages, are */i/*, */e/*, */a/*, */o/*, */u/*, and the most commonly encountered constants are */p/*, */k/*, */t/*. The vowel */a/* is believed to be the only phoneme that occurs in all languages.

An *allophone* is a phonetically distinct variant of a phoneme. The position of a phoneme within a word can cause it to be pronounced slightly differently; for instance, the */t/* sounds in *hit* and *tip* are allophones. Some languages do not distinguish between different pairs of phonemes; for instance, Japanese treats */l/* and */r/* as being the same phoneme, but different allophones.

Languages contain regularities in the ordering of phonemes. For instance, English words tend to follow a CVC structure (Consonant Vowel Consonant), and even within this structure certain patterns are heard much more frequently than others.[170]

syllable

The next higher-level unit of speech above a segment is known as a *suprasegmental*. The particular suprasegmental of interest to these coding guidelines is the syllable. A *syllable* consists of three parts: (1) the *onset*, (2) the *peak* or *nucleus*, and (3) the *coda*; for instance, for the syllable */man/*, */m/* is the onset, */a/* the peak, and */n/* the coda. A definition of syllable that readers might be familiar with is that of a consonant followed by a vowel (CV); however, this definition cannot be used to syllabify the structure CVCCV—is it CV-CCV or CVC-CV.

English syllables can have zero to three consonants before the vowel and zero to four consonants after the vowel. There has been relatively little research on automatically deducing how to divide a word into its component syllables.[332] Kessler and Treiman[170] investigated the structure of English syllables and found a correlation between a vowel and its following consonant. However, the correlation between a consonant and its following vowel was significantly lower, meaning the start of an English syllable is more informative (distinctive) than its ending.

The same syllables can be spoken at various levels of intensity. Varying the intensity, when speaking, enables certain parts of what is being said to be *stressed*. A languages stress pattern is of interest to these

abbreviating identifier

coding guidelines because it can affect how words are abbreviated. Different languages stress words in different ways. The difference between stressed and unstressed syllables in English[108] is greater than most other languages and it is common for the odd-numbered syllables to be stressed, with the first receiving the most stress (in prefixed words the primary stress usually falls on the first syllable of the root[52]). The same word may be stressed differently in different languages: English *GRAMmar* (from the French *gramMAIRE*)

morpheme

and *CHOColate* (from the Spanish *chocoLAte*).

The *morpheme* is the smallest grammatical unit of speech. There are two basic types; a *bound morpheme* is defined in terms of how it is attached to the other form, the *free morpheme*. The most common bound morphemes are prefixes and suffixes (e.g., *re-* and *-ed*; see Hawkins and Gilligan[136] for a discussion of

morphology

prefixing and suffixing universals).

The term *morphology* refers to the study of word structure and formation—the syntax of words. Pirkola[251] discusses the morphology of the world's languages from the perspective of information retrieval. Bauer[18] and Selkirk[273] discuss English word formation.

With the exception of logographic writing systems, there is some degree of correspondence between the

spelling

written and spoken form of a word (spelling is discussed elsewhere). Although C source is not designed to be a spoken language, many developers create a spoken form for it. The norms for this spoken form are most noticeable when they are broken (e.g., self-taught C programmers do not always have the opportunity to listen to C being spoken and invent their own pronunciations for some operators).

3.2.1 Speech errors

Cutler[79] provides a review of speech error data. A systematic, cross-linguistic examination of speech errors in English, Hindi, Japanese, Spanish, and Turkish by Wells-Jensen[323] found (reformatting a quote from her paper):

- *Languages are equally complex.* No overall differences were found in the numbers of errors made by

speakers of the five languages in the study. This supports the basic assumption that no language is more difficult than any other.

- *Languages are processed in similar ways.* Fifteen English-based generalizations about language production were tested to see to what extent they would hold true across languages. It was found that, to a large degree, languages follow similar patterns. For example, all of the languages exhibited the same pattern of semantically-based errors in open-class words, and all exhibited more errors with inflectional than derivational affixes. It was found, however, that the relative numbers of phonological anticipations and perseverations in other languages did not follow the English pattern.

- *Languages differ in that speech errors tend to cluster around loci of complexity within each.* Languages such as Turkish and Spanish, which have more inflectional morphology, exhibit more errors involving inflected forms, while languages such as Japanese, with rich systems of closed-class forms, tend to have more errors involving closed-class items.

3.2.2 Mapping character sequences to sounds

characters mapping to sound

[] Word recognition models of

It is believed that there are two possible routes by which readers convert character sequences to sound, (1) memory lookup, a direct mapping from the character sequence to sound; and (2) grapheme-to-phoneme conversion. Irregularly spelled words have to use the first route; for instance, the phrase "pint of water" does not contain any words that rhyme with *mint*. A study by Monsell, Patterson, Graham, Hughes, and Milroy[223] asked subjects to name lists of words, nonwords, or a mixture of both. The results showed that when nonword lists contained a few words, subjects tended to regularize (use the second route) the pronunciation of the words (the error rate was double that for lists containing words only).

A study by Andrews and Scarratt[9] investigated how people pronounce nonwords. For instance, would subjects pronounce *jead* using the regular grapheme-to-phoneme correspondence heard in *mead*, or would they use the irregular form heard in *head*? The results showed that 90% of pronunciations followed regular grapheme-to-phoneme rules. This is not to say that the pronunciations used for a particular word were always unique. In 63% of cases a nonword received one or two pronunciations, while 9.7% of nonwords received more than four different pronunciations. There seemed to be a common factor for nonwords where an irregular pronunciation was used; these nonwords did not have any regular word-body neighbors. This latter result suggests that perhaps speakers do not use grapheme-to-phoneme rules to build the pronunciations used for nonwords, but rather base it on the pronunciation of words they know that have similar spellings.

[782] neighborhood identifier

A study by Gibson, Pick, Osser, and Hammond[119] found that significantly more different pronunciations were used by subjects for letter sequences having a first-order approximation to English than letter sequences having a higher-order approximation. Deshmukh[87] discusses using maximum likelihood estimators to generate (potentially) multiple pronunciations, each with an associated likelihood probability.

Text-to-speech conversion is a very active research area with a growing number of commercial applications. To obtain high-quality output most systems rely on a dictionary of word-to-sound mappings. A number of rule-based algorithms have been proposed for converting letter sequences (graphemes) to sounds (phonemes).

Early work on creating a grapheme-to-phoneme mapping for a language involved a great deal of manual processing. For instance, Berndt, Reggia, and Mitchum[22] manually analyzed 17,310 words to derive probabilities for grapheme-to-phoneme mapping of English. Automating this process has obvious advantages. Daelemans and van den Bosch[80] created a language-independent conversion process that takes a set of examples (words and their corresponding phonetic representation) and automatically creates the grapheme-to-phoneme mapping. More recent research, for instance Pagel, Lenzo, and Black,[240] has attempted to handle out of vocabulary words (i.e., not in the training set); however, the quality of the conversion varies significantly. An evaluation of publicly available algorithms,[84] using a 16,280 word dictionary, found

correct conversion rates of between 25.7% and 71.8%.[7]

Divay and Vitale[90] provide a discussion of recent algorithms for grapheme–phoneme conversion of English and French, while Peereman and Content[244] provide quantitative information on the regularity of the mapping between orthography and phonology in English and French.

A category of letter sequences that often does not have the characteristics of words are people's names, particularly surnames. Correct pronunciation of people names is important in many applications and algorithms have been designed to specifically handle them. What can be learned from attempts to convert the written form of people's names to sound? A comparison of eight name-pronunciation systems[122] (two of which were human) found that acceptable (400-name test set, as judged by a panel of 14 listeners) performance varied from 78% to 98% for high-frequency names to 52% to 96% for very low-frequency names. Many of the systems tested included a pronunciation dictionary of several thousand to handle common cases and to help supplement the rules used (a collegiate-level English dictionary usually lists approximately 250,000 words). A list of unique surnames (in the USA) contains more than 1.5 million entries. To cover 50% of the surnames in the USA requires 2,300 names, while 50% of ordinary words can be covered in 141 words).

Vitale[320] and Llitjós[198] found that by taking into account the language of origin of proper names (statistical properties of the letter sequences in a name have been found to be a good indicator of its language of origin) it was possible to improve the accuracy of the phoneme transcription.

The Soundex algorithm[148] is often mentioned in introductory textbooks, which discuss sounds-like. This algorithm converts a word to a code consisting of the first letter of the word followed by up to three digits (obtained by looking up a value between zero and seven for subsequent letters). This algorithm has been used in a number of applications where a *sounds-like* capability is needed. Its very simplistic approach delivers results better than might be expected (a 33% success rate, with a 25% failure rate has been found for surnames[56]).

3.3 Words

Words do not exist in isolation; they belong to one or more human languages. Until relatively recently people experienced words in spoken form only. A number of studies[195] have found significant differences between spoken and written language usage. Because of the relatively high cost of analyzing spoken words compared to analyzing (invariably automated, using computers) written words, there has been significantly more published research on the written form. Source code is a written language and the studies quoted in this coding guideline section have primarily been of written text.

Many languages allow new words to be created by joining together, *compounding*, existing words. Other techniques used to create words include *prefixation* (*sleep* ⇒ *asleep*, *war* ⇒ *miniwar*) and *suffixation* (*kitchen* ⇒ *kitchenette*, *sea* ⇒ *seascape*).

agglutinative languages

In agglutinative languages, such as Japanese, the definition of a word is not always clear-cut. Words can be built up, like beads-on-a-string, using a series of affixes to the *root* word (the word that appears in a dictionary). There may also be letter changes at morpheme for phonetic reasons. For instance,[238] the Turkish root word *uygar*, meaning civilized, might have suffixes added to build the single word (translated as "(behaving) as if you were one of those whom we not be able to civilize", written with the components separated by + rather being immediately adjacent):

morpheme 782

```
uygar+laş+tir+ama+yabil+ecek+ler+imiz+den+miş+siniz+cesine
```

Part of the process of learning a language is producing and understanding compound words. The rules used and the interpretation given varies across languages. A study by Boucher[30] investigated the problems French students have in forming English compound words. He found that students made a number of different mistakes: used more than two terms, used incorrect affixes, pluralized the first word, or applied the

[7] Some researchers quote phoneme-conversion accuracy on a per letter-basis, while others use a per phoneme basis. A 90% per letter conversion accuracy equates to a $0.9^6 = 53\%$ word-conversion accuracy (for a six-letter word).

French rules. For instance, when asked the word for "a dog which hunts birds", answers included *dog-bird* and *bird-dog-hunting*. A later project by Boucher, Danna, and Sébillot[31] produced an intelligent tutoring system to teach students how to form English compound words.

While there might be general agreement on the pattern of usage of common compound words and phrases, there can be significant variation for rarer combinations. Technical terms often show variation between disciplines. One study[81] found 15 different forms of the term *Epithelial cell*.

The formation of compound words in English is discussed in more detail elsewhere. [] compound word

3.3.1 Common and rare word characteristics

Are there any differences in the characteristics of common and rare words?

An often-quoted study[192] investigated whether there were any differences in characteristics between high- and low-frequency English words (apart from their frequency of occurrence). The analysis suggested that differences existed; however, the analysis used four-letter words only, and words at the two frequency extremes. A later study by Frauenfelder, Baayen, Hellwig, and Schreuder[103] performed a more extensive analysis, for English and Dutch, using words containing between three and eight characters, with a range of frequencies.

Several of the Landauer et al. results (e.g., neighborhood density is higher for high-frequency words) were replicated for the case of four-letter English words. However, their findings did not hold across all word lengths. The results, for Dutch words, showed a weak but significant correlation between neighborhood density and word frequency. Although other differences were found (both for English and Dutch words), the authors were not able to find any significant word-frequency effects that applied across more than a few words lengths. [782] neighborhood identifier

A study by Howes and Solomon[150] found that the time taken to identify a word was approximately a linear function of the log of its relative frequency (the 75 most common words in the Thorndike–Lorge word counts were used).

3.3.2 Word order

Identifier names are sometimes built up from a sequence of words corresponding to a phrase or short sentence in a natural language familiar to the developer. Given a developer's natural language experience, the order of these words is likely to be the one that has semantic meaning in that language.

A few natural languages permit arbitrary word order, but in most cases the order used will have a semantic significance in the language used. For instance, the English sentence "Tanya killed Masha" has a different meaning than "Masha killed Tanya", while the words in the equivalent Russian sentence "Tanja ubila Mašu" could appear in any of the six permutations of the three words and still be a grammatically valid sentence with the same meaning. (However, the order SVO is the most frequently used in Russian; other languages having free word order also tend to have a frequently used order.)

The three principle components of a natural language sentence are the *object*, *subject*, and *verb*. While the order in which these three components occur can vary within a single language, most languages have a preferred order (the frequency with which different orders occur within a particular language will depend on whether it uses a pragmatic word order—*topic-prominent* such as in Chinese and Japanese—or uses a grammatical word order —*subject-prominent* such as in English and Spanish). Much of the material in this subsection is derived from *Language Universals and Linguistic Typology* by Bernard Comrie.[65]

Table 7: Known number of languages commonly using a particular word order. Based on Comrie.[65]

Common order	Languages	Example
None	no figures	Sanscrit
SOV	180	Turkish “Hansan ököz-ü aldı” ⇒ “Hassan ox bought”
SVO	168	English “The farmer killed the duckling”
VSO	37	Welsh “Lladdodd y ddraig y dyn” ⇒ “killed the dragon the man”
VOS	12	Malagasy “Nahita ny mpianatra ny vehivavy” ⇒ “saw the student the woman”
OVS	5	Hixkaryana “Toto yahosi-ye kamara” ⇒ “man it-grabbed-him jaguar”
OSV	2	Apurinã none available

There are three other major word-order parameters, plus many minor ones. The following are the major ones:

- Within a sentence a noun may be modified by an adjective (becoming a *noun phrase*). The two possible word orderings are AN (English “the green table”) and NA (French “le tapis vert” ⇒ “the carpet green”). Languages that use the order NA are more tolerant of exceptions to the rule than those using the order AN.
- A noun phrase may also contain a possessive (the technical term is *gentive*). The two possible word orderings are GN (Turkish “kadın-ın çavuğ-u” ⇒ “woman chicken-her”) and NG (French “la plume de ma tante” ⇒ “the pen of my aunt”). English uses both forms of possessive (the Saxon genitive “the man’s hat” and the Norman genitive “the roof of the house”). Although the Norman genitive is the more frequent, it is not possible to specify that it is the basic order.
- A language may contain prepositions, Pr (English “in the house”), or postpositions, Po (Turkish “adam için” ⇒ “man for the”).

There are 24 possible combinations of object/subject/verb, noun/adjective, noun/genitive, and preposition/postposition that can occur in languages. However, empirical data on existent languages shows the following common patterns:

- VSO/PR/NG/NA
- SVO/PR/NG/NA
- SOV/Po/GN/AN
- SOV/Po/GN/NA

word order adjectives

As well as using an order for different kinds of words, speakers also preferentially order the same kinds of words; for instance, the relative order in which adjectives occur.

Identifier concepts

3.4 Concepts

The extent to which the language used by a person influences their thought processes has been hotly debated over the centuries; more recently researchers have started to investigate how thought processes influence language use (see Lucy[203] for a detailed history). The proposal that language does influence thought is commonly known as the *Sapir-Whorf* or *Whorfian* hypothesis. Some people hold what is known as the *strong language-based* view, believing that the language used does influence its speakers’ conceptualization process. People holding the so-called *weak language-based* view believe that linguistic influences occur in some cases. The *language-as-strategy* view holds that language affects speakers performance by constraining what can be said succinctly with the set of available words (a speed/accuracy trade-off, approximating what needs to be communicated in a brief sentence rather than using a longer sentence to be more accurate).[155]

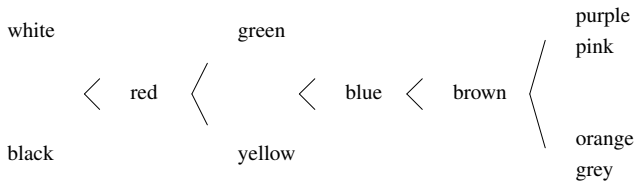


Figure 9: The original Berlin and Kay[21] language color hierarchy. The presence of any color term in a language, implies the existence, in that language, of all terms to its left. Papuan Dani has two terms (black and white), while Russian has eleven. (Russian may also be an exception in that it has two terms for blue.)

3.4.1 Metaphor

Metaphor

A data structure containing information about a politician's past record might include information about elections for which they have been a candidate. In the US politicians *run* for office, while in Spain and France they *walk*, and in Britain they *stand* for office. These are metaphors, and developers are likely to make use of them in the naming of identifiers (e.g., `ran_for`, `is_standing`).

Concepts involving time are often expressed using a spatial metaphor. These metaphors take two forms—one in which time is stationary and we move through it (e.g., "we're approaching the end of the year"); in the other case, we are stationary and time moves toward us (e.g., "the time for action has arrived").

A study by Boroditsky[28] investigated subject's selection of either the ego-moving or the time-moving frame of reference. Subjects first answered a questionnaire dealing with symmetrical objects moving to the left or to the right. The questions were intended to prime either an ego-moving or object-moving perspective. Subjects then read an ambiguous temporal sentence (e.g., "Next Wednesday's meeting has been moved forward two days"). The results found that 71.3 subjects responded in a prime-consistent manner. Of the subjects primed with the ego-moving frame, 73.3% thought the meeting was on Friday and 26.7% thought it was on Monday. Subjects primed with the object-moving frame showed the reverse bias (30.8% and 69.2%).

For a readable introduction to metaphors in everyday English see Lakoff and Johnson.[189]

3.4.2 Categories

Studies of color categorization provide a good example of the interaction between how peoples bodies work (in this case the eye), the category members supported by a language (in this case the different basic color terms), and human perception (see Hardin and Maffi[133] for an up-to-date discussion).

It was once thought that, across languages, color categories were arbitrary (i.e., the color terms used by different languages divided up the visible spectrum differently). In a now-classic study of 98 languages Berlin and Kay[21] isolated what they called the *basic color terms*. While the boundaries between color terms varied, the visual appearance of the basic color terms was very similar across languages (color matching has been found to be driven by physiological factors in the eye). They also found that the number and kind of basic color terms in languages followed a consistent pattern (Figure 9).

A survey of empirical behavioral and linguistic uses of color term studies by Corbett and Davies[72] found (languages studied included English, Russian, Japanese, French, Hebrew, and Spanish) that:

- time taken to name a color was faster for the basic color terms;
- when asked to name a color, the basic color terms were usually listed first;
- the rank frequency of basic color terms in text matched that predicted by Berlin and Kay (more occurrences of the more basic color terms).

English is the *world language* of business (and software development) and a study of Japanese by Stanlaw[289] found that English color loan words were supplanting the native ones (in an inverse order to the Berlin and Kay sequence).

words English

3.5 English

Languages have evolved complex processes for expressing subtle ideas. Although these processes may be obvious to native speakers, they tend to be unappreciated by inexperienced non-native speakers. This subsection uses English as an example for a discussion of some of the complexities of a human language. For a detailed discussion of English word formation, see Bauer[18] or Selkirk.[273] Bauer[19] discusses international varieties of English, as used by native speakers. Pinker[250] provides a readable discussion of word rules, taking as his subject regular and irregular verbs.

implicit learning

Many of the points covered in the following subsections will be familiar to those developers who speak English as a native language (although in some cases this knowledge will be implicit). The intent is to show to these developers the complexities of the language constructs they take for granted. Knowledge of, and practice using, these complexities takes many years of practice. Native speakers acquire it *for free* while growing up, while many non-native speakers never acquire it.

English optimized for

Analysis of the properties of English words suggest that they are optimized for recognition, based on their spoken form, using on their initial phonemes.[79, 274, 315]

phoneme 782

Technically, the terms *consonant* and *vowel* refer to spoken sounds—phonemes. In the written form of English individual letters are not unambiguously one or the other. However, the letters *a*, *e*, *i*, *o*, and *u* often represent vowel sounds. In some contexts *y* represents a vowel (e.g., *nylon*) and in some contexts *u* does not represent a vowel (e.g., *quick*).

compound word

3.5.1 Compound words

The largest group of compounds is formed using two nouns; noun+noun ⇒ stone wall, rainbow. (A study of technical terms in a medical database[81] found that 80% of unique multiword terms involved two nouns; see Costello[75] for a computational model.) Other compound forms include: verb+noun ⇒ *killjoy*, *spoilsport*; noun+verb ⇒ *homemade*, *rainfall*; adjective/adverb+noun ⇒ *quick-frozen*, *nearsighted* (see Costello[76] for a computational model); preposition+noun ⇒ *overload*, *underdog*; preposition+verb ⇒ *underestimate*, *overstep*; verb+particle ⇒ *makeup*, *breakdown*.

The creation and use of noun+noun compounds has been the subject of a number of studies. These studies have looked for restrictions that English speakers might place on permissible combinations (none found by Downing[91]) and the attributes associated with the individual words used to form the meaning of the new word.

A noun+noun combination has two parts in English. The first word acting as the modifier concept, while the second word is the head concept. Surveys[74, 327] have found the main kinds of combinations that occur are:

- *conjunctive*, where concepts from both words are combined; for instance, *pet bird* is a bird that is also a pet. These have been found to occur in less than 10% of compounds.

- *property*, where a property is transferred from one of the concepts to the other; for instance, an *elephant fish* is a big fish. Like relational interpretations, these have been found to occur between 30% to 70% of the time. The frequency of property combinations has been found to increase if the concepts are similar to each other.[326] For a study comparing the different theories of property combinations, see Costello.[77]

- *relational*, where a relation exists between two concepts; for instance, an *apartment dog* is a small dog that lives in city apartments. These have been found to be occur 30% to 70% of the time.

Words used to form an identifier might be a noun phrase rather than a compounded word. In this case word order differences may exist among dialects of English; for instance, British English uses *River Thames*, while American English uses *Hudson River*.

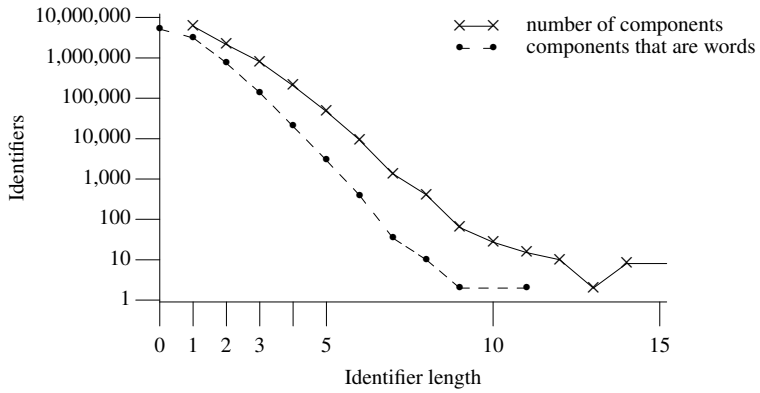


Figure 10: Number of *components* of an identifier (where a component is defined as a character sequence delimited by one or more underscore characters, _, the start of the identifier, or its ending, e.g., the identifier `big_blk_proboscis` is considered to contain three components, one of which is a word). A word is defined by the contents of the `ispell` 65,000 word list (this means, for instance, that the character sequence proboscis is not considered to be a word). Based on the visible form of the `.c` files.

3.5.2 Indicating time

Tense is the linguistic term given to the way languages express time. In English tense is expressed using a verb and is divided into three zones: past, present, and future (Table 8). Not all languages provide an explicit method of denoting the three tenses available in English; for instance, Japanese and Russian have no future tense.

Table 8: The 12 *tenses* of English (actually three tenses and four aspects). Adapted from Celce-Murcia.[53]

	Simple	Perfect	Progressive	Perfect progressive
Present	write/writes walk/walks	has/have written has/have walked	am/is/are writing am-is/are walking	has/have been writing has/have been walking
Past	wrote walked	had written had walked	was/were writing was/were walking	had been writing had been walking
Future	will write will walk	will have written will have walked	will be writing will be walking	will have been writing will have been walking

Time concepts that can be expressed in some other languages include distinctions between now versus not now and approximate versus not approximate.

3.5.3 Negation

English negation

The meaning of adjectives and adverbs can be inverted by adding a prefix (or sometimes a suffix). The following examples are from Celce-Murcia:[53]

```
happy ⇒ unhappy
appropriate ⇒ inappropriate
possible ⇒ impossible
logical ⇒ illogical
relevant ⇒ irrelevant
ordered ⇒ disordered
typical ⇒ atypical
life ⇒ lifeless
sense ⇒ nonsense
body ⇒ nobody
like ⇒ dislike
```

un- does not always indicate negativity; sometimes it indicates reversal (e.g., *unwrap*, *unfasten*). There are also words that do not follow this pattern (e.g., *inflammable* and *flammable* have the same meaning).

A noun can be negated by either adding *non-* or creating a non-phrase; for instance, *alternative* ⇒ *no alternative* and *sugar* ⇒ *sugar free*.

English also has two word forms that can be used to form a negation (i.e., not-negation and no-negation). Use of negation is much more common in speech that written text[24] (conversation is interactive and speakers have the opportunity to agree or disagree with each other, while written material usually represents the views of a single person). Studies of English grammar usage[24] have found that no-negation can be replaced with not-negation approximately 80% of the time (replacement in the other direction is possible in approximately 30% of cases), and that use of not-negation is several times more common than no-negation.

operand type ! The use of negation in C expressions is discussed elsewhere.

3.5.4 Articles

The English articles are: *definite* (e.g., *the* in "the book" would normally be taken to refer to a specific book), *indefinite* (e.g., *a*/*an* in "a book" does not refer to a specific book; the unstressed *some* is used for plural forms), and use of no article at all.

Most Asian and Slavic languages, as well as many African languages have no articles, they use article-like morphemes, or word order to specify the same information (e.g., the topic coming first to signal new information).

Experience shows that inexperienced users of English, whose native language does not have articles (e.g., Russian), have problems using the appropriate article. For instance, saying "have you book?" rather than"have you the book?" or "do you have a book?".

word order adjectives

3.5.5 Adjective order

In English adjectives precede the noun they modify. This is not always true in other languages. For instance, in French adjectives relating to age, size, and evaluation precede the noun, while adjectives referring to color or origin follow it:

```
une grande voiture jaune
    (big)   (car)  (yellow)
une vielle femme Italienne
    (old) (woman) (Italian)
```

Although it is rare for more than two adjectives to modify the same noun, the relative position of many of them has been found to have a consistent ordering. Svatko[298] used responses from 30 subjects to deduce a probability for the relative ordering of certain kinds of adjectives (Table 9).

Table 9: Probability of an adjective occurring at a particular position relative to other adjectives. Adapted from Celce-Murcia.[53]

determiner	option	size	shape	condition	age	color	origin	noun
	0.80	0.97	0.66	0.79	0.85	0.77	1.0	
an	ugly	big	round	chipped	old	blue	French	vase

3.5.6 Determine order in noun phrases

Within a noun phrase, determiners follow a general order; for instance:

```
pre core post
All our  many hopes . . .
core  post post
These next two weeks . . .
```

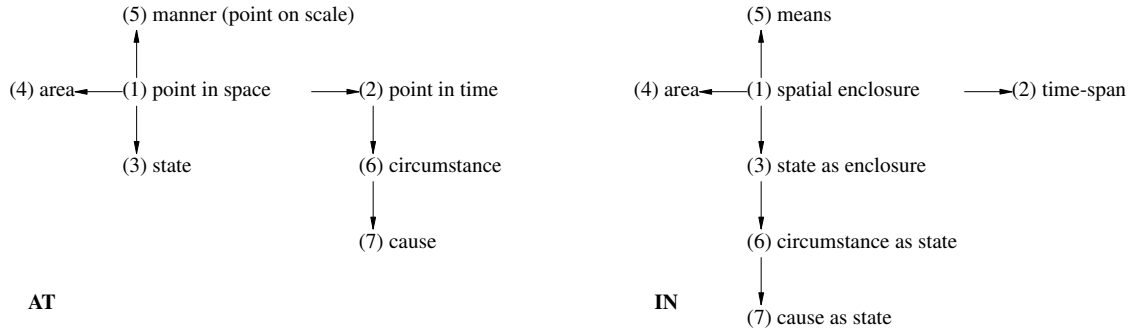


Figure 11: Examples, using "**at**" and "**in**" of extensions of prepositions from physical to mental space. Adapted from Dirven.[89]

Table 10: Subcategories of determiners. Adapted from Celce-Murcia.[53]

Predeterminers	Core determiners	Post determiners
qualifiers: *all, both, half*, etc. fractions: *such a, what a*, etc. multipliers: *double, twice, three times*, etc.	articles: *a, an, the*, etc. possessives: *my, our*, etc. demonstratives: *this, that*, etc. quantifiers: *some, any, no, each, every, either, neither, enough*, etc.	cardinal numbers: *one, two*, etc. ordinal numbers: *first, second*, etc. general ordinals: *next, last, another*, etc. quantifiers: *many, much, (a) few (a) little, several, more, less most, least*, etc. phrasal quantifiers: *a great deal, of, a lot of, a good number of*, etc.

Like adjective order, speakers of English as a second language often have problems using the appropriate determiner in the correct word order.

3.5.7 Prepositions

Prepositions

Prepositions are used to show role relationships. In some languages (e.g., Japanese) prepositions appear after the noun, in which case they are called *postpositions*. The same task is performed in some other languages (e.g., German, Russian) through the use of inflections.

A single preposition can express a wide range of relationships (it said to be *polysemous*); for instance, the networks in Figure 11 highlight the relationships between the following phrases:

1. Point in space: "**at** the station", or spatial enclosure: "**in** the station"
2. Point in time: "**at** six o'clock", time-span: "**in** one hour"
3. State: "**at** work", or "**in** search of"
4. Area: "good **at** guessing", or "rich **in** coal"
5. Manner: "**at** full speed", or "**in** a loud voice"
6. Circumstance: "**at** these words (he left)", or "she nodded **in** agreement"
7. Cause: "laugh **at**", or "revel **in**"

Tyler and Evans[309] give a detailed analysis of the range of meanings associated with spatial particles and provide a detailed analysis of the word *over*.[308] Allen[2] gives a temporal algebra that can be used to describe intervals of time.

English spelling

3.5.8 Spelling

English spelling has many rules and exceptions. Proposals to regularize, or simplify, the spelling of English have more than 200 years of history.[318] Experienced writers have learned to apply many of these rules and exceptions. For instance, the letter *e* is required after certain letters (*give*, *freeze*), or it may modify the pronunciation (*mat* vs. *mate*, *not* vs. *note*), or help to indicate that a word is not plural (*please*, *raise*), or it can indicate a French or Latin origin of a word (*-able*, *-age*).

A study by Venezky[319] of the 20,000 most frequent words led him to create his *seven principles of English orthography*, covering the correspondence from writing to sounds.

1. *Variation is tolerated.* Words can have alternate spellings, either across dialects (American *honor* versus British *honour*) or within a single community (*judgment* vs. *judgement*).
2. *Letter distribution is capriciously limited.* Only a few of the letter combinations that are possible are permitted. For instance, doubling is prohibited for the letters a, i, h, v, z (there are a few exceptions like *skivvy*).
3. *Letters represent sounds and mark graphemic, phonological and morphemic features.*
4. *Etymology is honored.* That is, spelling relates to the history of English. For instance, a word that was borrowed early from French will have a */tS/* correspondence for *ch* (e.g., *chief*), while a word that was borrowed from French at a later period will have a */S/* correspondence (e.g., *chef*).
5. *Regularity is based on more than phonology.* For instance, there is the so-called *three letter rule*. All one- or two-letter words are function words[8] (e.g., *I*, *by*, *to*, *an*, *no* etc.), while content words have three or more letters (e.g., *eye*, *bye*, *two*, *Ann*, *know*).
6. *Visual identity of meaningful word parts takes precedence over letter-sound.* The claims that English spelling is illogical are often based on the idea that spelling should correspond to speech sounds. However, English spelling attempts to represent the underlying word forms (stripped of features attached to them by phonological rules), not represent sounds. English uses a lexical spelling system: one spelling, one morpheme, whatever the permutations of pronunciation; for instance, *cup*/*cupboard*, *critic*/*criticise*, *like*/*liked*/*likes*, *sign*/*signature*,
7. *English orthography facilitates word recognition* for the initiated speaker of the language rather than being a phonetic alphabet for the non-speaker.

In English the pronunciation of a particular sequence of letters can depend on their position in the word. For instance, the letter sequence *ghoti* could be pronounced as *fish* (*gh* as in *cough*, *o* as in *women*, and *ti* as in *nation*).[9]

When experienced English speakers are asked to spell spoken words, they are sensitive to the context in which the vowels occur,[307] the position of the phoneme within the word,[246] and other idiosyncratic factors.

identifier
English as second language

3.6 English as a second language

English is not only the *world language* of business, but also of software development. Information encoded in an identifier's spelling can only be extracted by readers if they are familiar with the English grammar, or English words, that it makes use of. Developers writing source that will include a non-native speaker of English readership might want to consider the benefits of restricting their usage of English to constructs likely to be familiar to this audience.

Even in those cases where developers appear to have near-native English-speaking ability there may be significant differences in their grammar usage.[71] Grammar plays a role in identifier spelling because,

[8] There are some rarely used words that are exceptions to this rule *e.g., *ox*, *ax* a US and old English spelling of *axe*) and specialist words such as *id*.

[9] George Bernard Shaw's original observation referred to the possible spelling of the sound */fish/*.

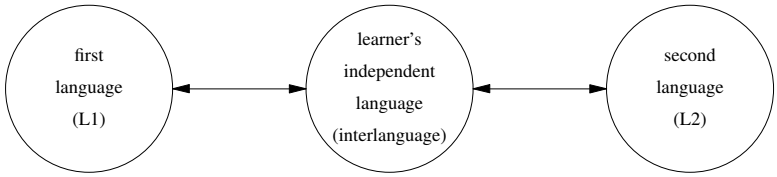


Figure 12: A learners independent language—*interlanguage*. This language changes as learners go through the various stages of learning a new language. It represents the rules and structures invented by learners, which are influenced by what they already know, as they acquire knowledge and proficiency in a new language.

in English, words may have forms that reflect their grammatical role within a sentence. The learning of grammatical morphemes (e.g., *-ing*, *-s*) has been found to occur (in children and adults) in a predictable sequence.[123] The following list is based on Cook,[68] who also provides a good introduction to the linguistic factors involved in second-language teaching:

1. plural *-s* (e.g., "Girls go")
2. progressive *-ing* in present continuous form (e.g., "Girls going")
3. copula forms of *be* (e.g., "Girls are here")
4. auxiliary form of *be* (e.g., "Girls are going")
5. definite and indefinite articles—*the* and *a* (e.g., "The girls go" or "A girl go")
6. irregular past tense (i.e., verbs that do not have the form *-ed*)—(e.g., "The girls went")
7. third person *-s* (e.g., "The girl goes")
8. possessive *s* (e.g., "The girl's book")

The ordering of this sequence does not imply an order of difficulty in learning to use a construct. Studies[194] have found some significant variation in the ease with which learners acquire competence in the use of these constructs.

How many words does a speaker of English as a second language need to know? English contains (per *Webster's Third International Dictionary*, the largest dictionary not based on historical principles) around 54,000 word families (*excited*, *excites*, *exciting*, and *excitement* are all part of the word family having the headword *excite*). A variety of studies[231] have shown that a few thousand word families account for more than 90% of words encountered in newspapers and popular books. A variety of basic word lists,[231] based on frequency of occurrence in everyday situations, have been created for people learning English. All application domains have their own terminology and any *acceptable use* list of words will need to include these.

Non-native speaker's ability to extract information from identifiers created by native speakers may currently be the primary commercial developer language concern. However, the amount of source written (and therefore identifiers created) by non-native speakers continues to grow. The importance of native speakers, and speakers having a different first language to the original developer, to extract information from identifiers created by non-native speakers will grow as the volume of code increases. Handling different developer *interlanguages*[299] (Figure 12) is likely to be difficult.

There are lower-level reading issues associated with developers who have English as a second language, including (see Henser[139] for a survey of research on language-specific thoughts about bilinguals, and Carlo and Sylvester[49] for research on second-language reading):

- A study by van Heuven, Dijkstra, and Grainger[313] found that the orthographic neighborhood of both languages affected the performance of bilinguals. [782] neighborhood identifier

- A study by Ziegler, Perry, Jacobs, and Braun[333] investigated how identical words and nonwords (in some cases there were minor spelling differences) were read in English and German. They found that body neighborhood had a larger impact on naming time in English than German. The error rates were consistent across the various conditions tested at 1.8% and 3.7% for English and German, respectively.
- A study by Malt and Sloman[209] asked second language users of English to name household objects (e.g., bottles, jars, plates, and bowls). They found that the names given by subjects did not fit the categorization patterns of native English speakers (subject performance was found to depend on the number of years of experience using English in a non-classroom setting). In the same way that the name applied to an object, by a native speaker, can depend on the context in which it occurs, there are cultural differences that affect the name given for an object.

context naming affected by naming 782 cultural differences

4 Memorability

identifier memorability memory developer

An overview of human memory was given in Sentence 1. To summarize, short-term memory is primarily sound-based (some studies have found a semantic component), while long-term memory is on semantics-based (meaning).[10]

This section discusses specific issues, those relating to identifiers, in more depth. In this coding guideline section memorability refers to the different ways developers recall information associated with an identifier in the source code. The kinds of information and their associations include:

- The spelling of one or more identifiers may need to be recalled. Spelling recall is usually indexed by semantic rather than letter associations. These associations may relate to information denoted by the identifier (e.g., the kind of information held in an object), its usage in the source code (e.g., a loop variable or the names of file scope objects modified by a call to a function), or some unique aspect of its declaration (e.g., a single parameter or label in a function definition).
- The information denoted by an identifier need may need to be recalled. This recall is usually indexed by the character sequence of an identifier's spelling.
- A previously seen identifier may need to be recognized as denoting the same entity when it is encountered again while reading source code.
- The locations in the source code that reference, or declare, an identifier may need to be recalled. This recall may be indexed by information on spelling or semantic associations.
- All the members of a list of identifiers may need to be recalled. The indexing information used for the recall and the kind of information required to be recalled varies with the identifier list and the context in which it is referenced. For instance, writing a **`switch`** statement whose controlling expression has an enumerated type requires knowledge of the names of the enumeration constants for that type, and use of designators in an initializer requires knowledge of the names of structure members. However, while a function invocation requires information on the expected arguments, that information is associated with the name of the function and the names of any parameters are rarely of interest.

designator . identifier

In most cases the required information is available in the source code (using identifiers in third-party libraries is one exception). However, readers would need to invest resources in locating it. The trade-offs people make when deciding whether to invest resources in locating information or to use information in their heads is discussed elsewhere.

cost/accuracy trade-off identifier developer interaction

The different kinds of developer interaction with identifiers places different demands on human memory

[10] Designers of IDEs ought to note that at least one study[234] found that highlighting parts of a word does not produce any improvement in recall performance.

resources. For instance, identifiers defined within functions are often referred to, by developers, as being *temporary* (a temporary object, temporary storage, or just a temporary). For a reader of the source the time interval over which they are *temporary* is the time needed to obtain the required information about the function being read. In the case of functions containing a few lines of code this might only be a few seconds, but in most cases it is likely to be more than a minute. During a day's work reading source code, a developer is likely to read many function definitions, each containing zero or more of these *temporary* identifiers, and some possibly sharing the same spelling.

What is needed is the ability to be able to recall them while reading the body of the function that declares them and then to forget about them after moving on to the next function definition. Rather like walking out of a supermarket and recalling where the car is parked, but not being confused by memories for where it was parked on previous visits. Unfortunately, people do not have the ability to create and erase information in their memory at will.

While the brain mechanisms underlying human memory is a very active research area, it is not yet possible to give a definitive answer to any fundamental questions (although some form of semantic network for connecting related individual concepts is often used in modeling). Without a reliable model, it is not possible to describe and predict memory performance, and this section approaches the issue by trying to draw conclusions based on the various studies involving words and nonwords, that have been performed to date.

[] semantic networks

The following subsection discusses some of the studies that have been performed on human recall of different kinds of names (e.g., proper names and common names) and the various research projects aimed at finding out how people make the connection between a name and what it denotes. This is followed by three subsections that discuss the issues listed in the preceding three bullet points.

4.1 Learning about identifiers

It might be thought that developers would make an effort to remember the names of identifiers; for instance, by reading the locally declared identifiers when first starting to read the source of a function definition. However, your author's experience is that developers often read the executable statements first and only read declarations on an as-needed basis. Developers only need information on locally declared identifiers while reading the source of the function that contains them. The cost of rehearsing information about locally declared identifiers to improve recall performance is unlikely to be recouped. Whether looking up identifier information on an as-needed basis is the optimal cognitive cost-minimization technique is an open question and is not discussed further here.

What information do developers remember about identifiers? There is no research known to your author addressing this question directly. The two most common important memory lookup queries involving identifiers are their spelling and semantic associations. Before these two issues are discussed in the last two subsections of this section on memorability, there is a discussion on the results from various memory studies and studies of people's performance with proper names.

Information about identifiers is provided by all of the constructs in which they occur, including:

- A declaration (which may include an associated comment) provides information on the type, scope, and various other attributes. It is likely that the reader will want to recognize this declared identifier later and recall this information about it.
- An expression will reference identifiers that need to be recognized and information about them recalled. Being referenced by an expression is also potentially useful information to be remembered, along with the identity of other identifiers in the same expression.

The issue of how readers represent identifier information associated with declarations and expressions in memory is discussed elsewhere.

[] declaration syntax
[] expressions

4.2 Cognitive studies

The power law of learning implies that the more often an identifier is encountered (e.g., by reading its

[] power law of learning

memory information elaboration

name or thinking about what it represents) the more likely it is to be correctly recalled later. Studies have also found that the amount of processing performed on to-be-remembered information can affect recall and recognition performance.

Human memory for pairs of items is not always symmetrical. For instance, a person who has learned to recall *B* when prompted with *B* might not recall *A* so readily when prompted with *B*. This issue is not discussed further here (see Kahana[166] for a discussion).

The performance of human memory can be depend on whether information has to be recalled or whether presented information has to be recognized. For instance, a person's spelling performance can depend on whether they are asked to recall or recognize the spelling of a word. A study by Sloboda[281] asked subjects to choose which of two words was correctly spelled. The results showed (Table 11) significantly more mistakes were made when the alternative was phonologically similar to the correct spelling (191 vs. 15); that is, the spelling looked sufficiently plausible.

Table 11: Example words and total number of all mistakes for particular spelling patterns (*–C–* denotes any consonant). Adapted from Sloboda.[281]

Spelling pattern	similar phonologically	mistakes made	dissimilar phonologically	mistakes made
-ent -ant	clement clemant	46	convert convart	1
-ce -se	promice promise	9	polich polish	1
w- wh-	weight wheight	3	sapely shapely	1
-er -or	paster pastor	7	parret parrot	6
-le -el	hostle hostel	11	assits assist	1
-ayed -aid	sprayed spraid	18	slayer slair	0
-ea- -ee-	deamed deemed	24	dearth deerth	3
-CC- -C-	deppress depress	33	preessed pressed	0
-ancy -ency	currancy currency	27	corractly correctly	0
-al -el	rival rivel	13	livas lives	2

identifier recall

4.2.1 Recall

initial letters identifier 782

The initial letters of a word are a significant factor in several word related activities (it has been suggested

identifier tip-of-the-tongue

- A study by Rubin[267] investigated the so-called *tip-of-the-tongue* phenomenon. People in the tip-of-the-tongue state know that they know a word, but are unable to name it. Rubin gave subjects definitions of words and asked them to name the word (e.g., somebody who collects stamps, a *philatelist*). Those subjects who knew the word, but were unable to name it, were asked to write down any letters they thought they knew. They were also asked to write down the number of syllables in the word and any similar-sounding words that came to mind. The results showed that letters recalled by subjects were often clusters at the start or the end of the word. The clusters tended to be morphemes (in some cases they were syllables). For instance, in the case of *philatelist* many subjects recalled either *phil* or *ist*.

identifier context cueing recall

- Context can also play an important role in cueing recall. A study by Barclay, Bransford, Franks,

McCarrell, and Nitsch[16] investigated how cued recall varied with context. Subjects were divided into two groups and each group was shown a different list of sentences. For instance, the list of sentences seen by the members of one group might include—"The secretary put the paper clips in the envelope", while the other group would see a different sentence relating to secretaries and envelopes "The secretary licked the envelope". After seeing the sentences, subjects heard a list of cues and were asked to write down the noun from the list of sentences each cue reminded them of. Examples of cues included "Something that can hold small objects" and "Something with glue". It was predicted that cues matching the sentence context would produce better recall. For instance, the cue "Something that can hold small objects" is appropriate to paperclips (small objects) and envelopes (being used to hold something), but not directly to an envelope being licked (where the glue cue would have a greater contextual match). The results showed that subjects hearing cues matching the context recalled an average of 4.7 nouns, while subjects hearing cues not matching the context averaged 1.5 nouns.

- The visual similarity of words can affect serial recall performance. A study by Logie, Sala, Wynn, and Baddeley[199] showed subjects a list of words that was acoustically similar (to reduce the possibility of phonological information being used to distinguish them), but one set was visually similar (e.g., *FLY*, *PLY*, *CRY*, *DRY*) while the other set was visually distinct (e.g., *GUY*, *THAI*, *SIGH*, *LIE*). The results showed that the mean number of words recalled in the visually similar list was approximately 10% lower, across all serial positions, than for the visually dissimilar list.

4.2.2 Recognition

identifier
recognition

Recognition is the process of encountering something and remembering that it has been encountered before. Recognition is useful in that it enables previously acquired information to be reused. For instance, a reader may need to check that all the operands in an expression have a given type. If an identifier occurs more than once in the same expression and is recognized when encountered for the second time, the information recalled can remove the need to perform the check a second time.

Failing to recognize a previously seen identifier incurs the cost of obtaining the needed information again. Incorrectly recognizing an identifier can result in incorrect information being used, increasing the likelihood of a fault being introduced. The first study below describes one of the processes people use to work out if they have encountered a name before. The other two studies discuss how semantic context effects recognition performance. See Shiffrin and Steyvers[278] for a recent model of word-recognition memory.

- A study by Brown, Lewis, and Monk[41] proposed that people use an estimate of a word's memorability as part of the process of deciding whether they had previously encountered it in a particular situation. For instance, names of famous people are likely to be more memorable than names of anonymous people. If presented with a list of famous names and a list of non-famous names, people are more likely to know whether a particular famous name was on the famous list than whether a non-famous name was on the non-famous list. The results of the study showed that in some cases name memorability did have an affect on subject's performance.
- A study by McDermott[216] asked subjects to memorize short lists of words. The words were chosen to be related to a nonpresented word (e.g., *thread*, *pin*, *eye*, *sewing*, *sharp*, and *thimble* are all related to *needle*). Subjects were then presented with words and asked to specify whether they were in the list they had been asked to memorize. The results showed that subjects recalled (incorrectly) the related word as being on the list more frequently than words that were on the list. The effect persisted when subjects were explicitly told not to guess, and a difference of 30 seconds or 2 days between list learning and testing did not change the observed pattern.
- A study by Buchanan, Brown, Cabeza, and Maitson[43] used a list of words that were either related to each other on a feature basis (e.g., *cat* and *fox* share the features: four legs, fur, tail, etc.) or by association (e.g., thread, pin, eye, sewing, thimble). The activation in a semantic network organized by features would be expected to spread to words that were related in the number of features they

shared, while in an associated organization, the activation would spread to words that were associated with each other, or often occurred together, but did not necessarily share any features. The results showed that for an associated words list, subjects were much more likely to falsely recall a word being on the list, than when a feature-based words list was used.

identifier phonetic symbolism

Recognition through the use of phonetic symbolism is discussed elsewhere.

Ranschburg effect

4.2.3 The Ranschburg effect

Some identifiers consist of a sequence of letters having no obvious pronunciation; for instance, `hrtmb`. In this case readers usually attempt to remember the individual letters of the sequence.

When the same letter occurs more than once in a letter sequence, a pattern of short-term memory behavior known as the *Ranschburg effect* occurs.[140] If occurrences of the same letter are separated by other letters, `hrtrb`, recall performance for the duplicate letter is reduced (compared to the situation where a nonduplicate letter appears at the same position in the sequence). If occurrences of the same letter occur together, `hrrtb`, recall performance for the duplicate letter is improved. This effect has also been found to occur for digits.[324]

identifier learning a list of

4.2.4 Remembering a list of identifiers

In many contexts a sequence of identifiers occur in the visible source, and a reader processes them as a sequence. In some cases the identifiers in the sequence have a semantic association with each other and might be referred to as a list of identifiers. In other cases the only association connecting the identifiers is their proximity in the source.

```
1  typedef int ZIPS;
2
3  enum e_t {aa, bb, cc, dd};
4
5  struct s_t {
6              int mem_1;
7              long ee;
8             } xyz;
9
10 void f(int p_1, float foo)
11 {
12 ZIPS average,
13      total;
14 int loc;
15 double bar;
16
17 bar=foo+xyz.ee-cc;
18 }
```

primacy effect memory recency effect memory identifier confusability identifier semantic associations

A number of factors have been found to be significant when information on a sequence of identifiers needs to be recalled or remembered. The primacy and recency effects, confusability, and semantic issues are discussed elsewhere.

The following study illustrates how some of these factors affect subject's performance.

A study by Horowitz[149] asked subjects to learn a list of 12 trigrams. One list, known as *L4* was created using the four letters F, S, V, and X, while another list, known as *L12*, was created from 12 different consonants. Because there were only four letters to choose from, the trigrams in the first list often shared one or more letters with other trigrams in the list. The trigrams in the second list were chosen so that a particular pair of letters occurred in only one item.

A trial consisted of showing the subjects one of the lists of 12 trigrams. One group was then asked to write down as many as they could freely recall, while a second group had to arrange slips of paper (each containing a single, presented trigram) into the same order as the presentation. Subjects' performance was measured after each of 10 trials (each using a different order of the 12 trigrams).

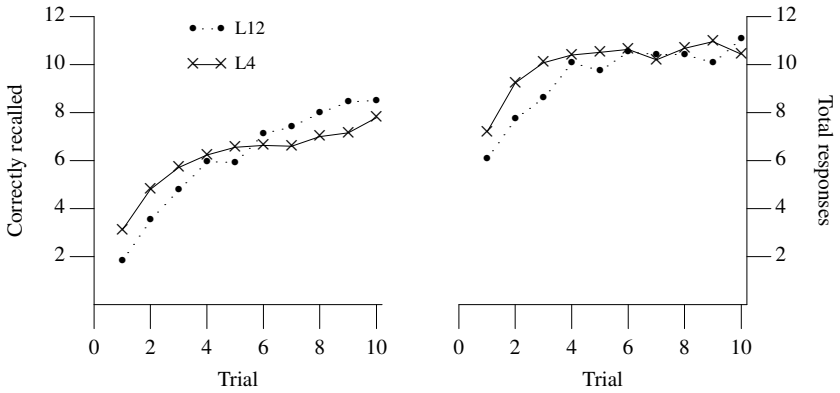


Figure 13: Mean correct recall scores and mean number of responses (correct and incorrect) for 10 trials. Adapted from Horowitz.[149]

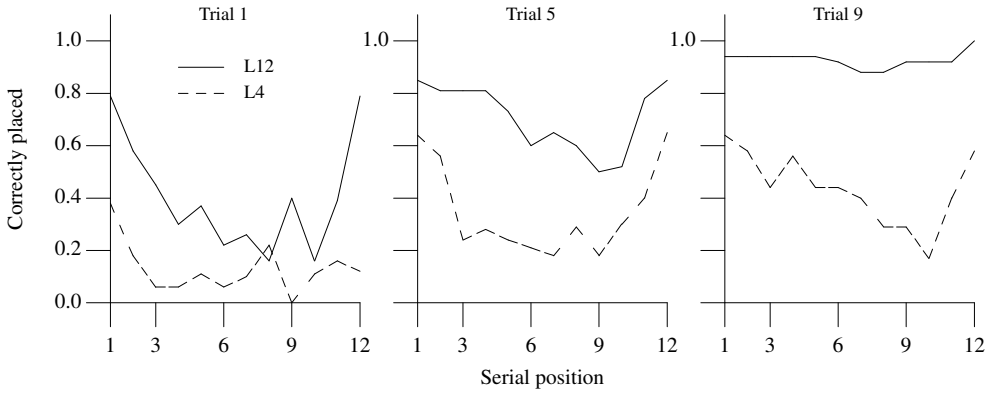


Figure 14: Percentage of correct orderings as a function of the trigram position within the list learned for three different trials. Adapted from Horowitz.[149]

The results for the free recall of trigrams (Figure 13) show that initially subjects working with the L4 list performed best (the probability of randomly combining four letters to produce a correct three-letter trigram is 50%; this difference may be due to the much higher probability of being able to randomly pick a correct answer for the L4 list, compared to the L12 list). With practice (approximately six trials) recall performance of the group working with the L12 list exceeded that of the group using the L4 list.

The results for the ordering of trigrams (Figure 14) show primacy and recency effects for both lists. The performance of subjects working with the L12 list is significantly better than those using the L4 list over all trials.

[] primacy effect memory
[] recency effect memory

A few studies have found that semantic information appears to be represented in short-term memory rather than simply being retrieved from long-term memory. A consequence of this representation is that semantic information associated with an identifier can affect recall performance. A study by Haarmann and Usher[130] showed subjects a list of six pairs of semantically related words and asked them to recall as many words as possible. In one list the semantically related words were adjacent to each other (e.g., "broader horizon king leader comfort uneasy kittens fluffy get purchase eating lunch"), while in the other they were separated by five unrelated words (e.g., "broader king comfort kittens get eating horizon leader uneasy fluffy purchase lunch"). The results showed that recall from the list of semantically adjacent words had an improved recency effect; that is, recall of words at the end of the list was better for semantically adjacent words than semantically separated words.

[] working memory information representation

[] recency effect memory

4.3 Proper names

A number of researchers have been investigating people's memory performance for proper names.[11] The results appear to show that the human brain does not handle different kinds of names in the same way. Studies of patients with aphasia (damage to the brain that causes problems with words) have found that different kinds of damage cause different kinds of problems. One study[125] found that some subjects had problems naming body parts but could name geographical locations, while others had problems naming geographical locations but could name body parts. Categories of names subject to these problems include animate versus inanimate objects, numbers versus letters, referential versus. descriptive, and common versus proper names. Another study[124] found that brand names were handled differently in the brain than proper names.

Proper name recall performance has been found to decrease with age,[60] although in part this may be due to differences in mnemonic strategies (or lack of strategies).[39]

Many people experience greater difficulty remembering people's names than remembering other kinds of names. A number of possible reasons for this behavior have been proposed, including:

- Proper names are unique, and no alternatives are available. A study by Brédart[36] showed subjects the faces of people having two well-known names (e.g., Sean Connery alias James Bond and Peter Falk alias Columbo) and people having a well-known and a little-known name (e.g., Julia Roberts is not closely associated with any character's name she has played). Subjects were *blocked* (i.e., no correct answer given) on those faces having a single well-known name in 15.9% of cases, but were only *blocked* for 3.1% of faces having two well-known names (giving either name counted as a non-blocked response).

- The rate of learning of new common nouns slows as people reach adulthood. Adults learn technical names through further education and work experience, but learning new names in general usage is rare. However, people continue to learn new proper names throughout their lives. Being introduced to a *Mr. Dreaner* would not be considered unusual, but being told that a person was a *dreaner* might lead the listener to conclude they had misheard the job description. The range of plausible phonologies[37] is much greater for proper names than for common names. Having a set of known common names makes it easier to guess a particular name, given limited information about it (e.g., the first few letters).

A study by McWeeny, Young, Hay, and Ellis[218] asked subjects to learn to associate various names and professions with pictures of unfamiliar faces. Subjects were then presented with each picture in turn and asked to recall the associated name and profession. In those cases where only one association could be recalled there was a significant probability that it would be the profession rather than the name. In some cases the labels used could be either a name or a profession (e.g., Baker or Potter). Subjects asked to associate these ambiguous labels with a profession were more likely to recall them than those subjects asked to associate them with a name. This has been termed the *Baker–baker* paradox. The difficulty of recalling people's names is not directly related to the features of the name, such as its frequency of occurrence.

A number of other studies[310] have confirmed that generally, recall of semantic properties associated with a person is faster than recall of that person's name. A study by Cohen[59] tested the hypothesis that the reason for this difference in performance was because peoples names are meaningless labels. There is no semantic association, in the subjects head, between the name and the person. In the first experiment subjects were asked to learn an association between a photograph, a name, an occupation, and either a meaningless nonword or a meaningful word (e.g., "Mr Collins; he is a teacher; he has a wesp", or "Mr Collins; he is a teacher; he has a boat").

[11] English, and most Indo-European languages, distinguish between two kinds of nouns. Names that denote individuals are called *proper names* (or *proper nouns*), while all other names (referring to classes of objects) are called *common names*.

Table 12: Mean number of each kind of information recalled in each condition (maximum score: 48). Adapted from Cohen.[59]

	Name	Occupation	Possession
Nonword	18.6	37.1	16.5
Word	23.6	37.0	30.4

The results (Table 12) show that in the nonword case recall of both names and possessions was similar (the slightly greater recall rate for names could be caused by a frequency effect, the nonword names being names familiar to the subjects). When words were used for names and possessions, the relative recall rate changed dramatically. Cohen points out that information on real-world possessions has many semantic associations, which will trigger more connections in the subject's lexical network. A word used as a person's name is simply a label and is unlikely to have any semantic associations to an individual.

To what extent can parallels be drawn between different kinds of source code identifiers and different kinds of natural language names? For instance, are there similarities between the way developers treat the members of a structure and body parts, or between the way they think about labels and geographical locations? Identifiers do not always consist of a single word or non-word; they can form a phrase (e.g., `total_time`). Your author has not been able to find any studies looking at how human memory performance varies between words and phrases.

The two conclusions that can be drawn from studies about the recall of names is that richness of semantic associations can improve performance and that it is possible for different kinds of names to have different recall characteristics.

4.4 Word spelling

spelling

Spelling is the process of generating the sequence of characters that are generally accepted by native speakers of the language to represent the written form of a particular word. Software developers have extended this definition to include source code identifiers, whose names are commonly said to have a *spelling*.

Typing an identifier on a keyboard involves using memory to recall the sequence of letters required (or a rule to derive them) and motor skills to type the character sequence. This section provides a general discussion on the studies that have been made of spelling. The motor activities associated with typing are discussed elsewhere.

[] typing mistakes

Research into spelling has been carried out for a number of reasons, including learning about cognitive processes in the brain, trying to improve the teaching of children's reading and writing skills, and the creation of automated spelling correction programs . A lot of this research has used English speakers and words. It is possible that the models and theories applicable to users of a language that has a deep orthography may not apply to one that has a shallow orthography (e.g., Spanish[165]). Given that the spelling of identifiers is often more irregular than English, the availability of so much research on irregular spellings is perhaps an advantage.

In some languages spelling is trivial. For instance, in Spanish if a person can say a word, they can spell it. Writers of other languages experience even more problems than in English. For instance, agglutinative languages build words by adding affixes to the root word. The effect of having such a large number of possible words (nouns have approximately 170 basic forms in Turkish and 2,000 in Finnish) on the characteristics of native speaker spelling mistakes is not known (automating the process of determining the root word and its affixes is a difficult problem in itself[238]).

If people make spelling mistakes for words whose correct spelling they have seen countless times, it is certain that developers will make mistakes, based on the same reasons, when typing a character sequence they believe to be the spelling of an identifier. The following subsections discuss studies of spelling mistakes and some of the theories describing readers spelling performance. The aim is to find patterns in the mistakes made, which might be used to reduce the cost of such mistakes when typing identifier spellings. The fact that an identifier spelling may contain words, which may be misspelled for the same reasons as when they occur in prose, is a special case.

Beware of heard,
a dreadful word,
That looks
like beard and
sounds like bird,
And dead:
it's said like
bed, not bead,
For Goodness'
sake, don't
call it deed!
Watch out
for meat and
great and threat,
They rhyme
with suite and
straight and debt.
– Anon

[782] agglutinative languages 55

4.4.1 Theories of spelling

How do people produce the spelling of a word? The two methods generally thought to be used are mapping from phonemes to graphemes (rule-based) and memory lookup (memory-based). The extent to which either of these methods is used seems to vary between people (those who primarily use the rule-based method are sometimes called *Phoenicians*, and those primarily using the memory-based method are called *Chinese*). A study by Kreiner and Gough[179] showed that good spellers made use of both methods.

Another possible method is that spelling is performed by analogy. A person uses a word for which the spelling is known which is phonemicly or graphemicly similar as a model, to produce the spelling of the unknown word.

4.4.2 Word spelling mistakes

The idea that there is a strong correlation between the kinds of spelling mistakes people make when writing prose and the kinds of spelling mistakes they make when writing C identifiers sounds appealing. Some broad conclusions about common prose spelling patterns can be drawn from studies of spelling mistakes (both written and typed). Kukich[184] provides a review of automatic spelling correction of text. However, the data on which these conclusions are based was obtained from disparate sources performing under very different conditions. Given the task and subject differences between these studies and developers writing code, any claims that these conclusions can be extended to developer performance in spelling identifiers needs to be treated with caution. For this reason the next subsection discusses the spelling mistake data, used in reaching these conclusions in some detail.

The following are some of the general conclusions drawn from the studies of spelling mistakes (a mistake is taken to be one of the operations insertion, deletion, substitution, or transposition):

- Between 69% to 94% of misspelled words contain a single instance of a mistake. The remaining misspelled words contain more than one instance.
- Between 1.4% to 15% of misspellings occurred on the first letter (this does not confirm the general belief that few mistakes occur in the first letter).
- Studies of typing performance have found strong keyboard adjacency effects (a letter adjacent to the one intended is typed). However, no spelling study has analyzed the effects of keyboard adjacency.
- There is insufficient data to analyze whether the number of mistakes made in a word is proportional to the number of letters in that word. (One study[179] found that the probability of a mistake being made increased with word length.)
- The pronunciation used for a word has a strong effect on the mistakes made. An incorrect spelling is often a homophone of the correct spelling.

Table 13: Breakdown of 52,963 spelling mistakes in 25 million typed words. Adapted from Pollock and Zamora.[255]

Kind of Mistake	Percentage Mistakes
omission	34
insertion	27
substitution	19
transposition	12.5
more than one	7.5

Many of the spelling mistake data sets are derived from words people have chosen to use. However, people tend to limit the words they use in written prose to those they know how to spell.[227] Developers often have to use identifiers created by others. The characteristics of spelling mistakes for words chosen by other people are of interest. An analysis by Mitton[220] looked at the differences in spelling mistakes made

by 15 year-old children in written prose and a spelling test. The results show that, compared to mistakes in prose, mistakes in spelling test words contain more multiple mistakes in longer words (longer words are used less often in prose because people are less likely to be able to spell them).

A study by Adams and Adams[1] asked subjects to either spell a word (generation) or to select one of three possible spellings (recognition). Subjects also had to rate their confidence in the answer given. The results showed subjects were underconfident at low confidence levels and overconfident at high confidence levels, a commonly seen pattern. However, subjects' estimates of their own performance was more accurate on the recognition task. [] overconfidence

Measuring spelling mistakes is not as straight-forward as it sounds. The mistakes made can depend on the subjects educational level (the educational level of many of the subjects at the time the data was obtained was lower than that typical of software developers, usually graduate-level), whether they or other people selected the words to be spelled, whether the mistakes were manually or automatically detected. Also, the English language accent, or dialect, spoken by a person has been found to affect word spelling performance for adults[306] and children.[26, 86, 291]

Like any other character, the space character can be mistyped. The difference between the space character and other characters is that it cannot occur within identifiers. An extra space character will cause a word to be split in two, which in C is very likely to cause a syntax violation. Omitting a space character is either harmless (the adjacent character is a punctuator) or will cause a syntax violation (in C). For these reasons, mistakes involving the space character are not discussed further here.

4.4.2.1 The spelling mistake studies

The spelling mistake studies and their findings are described next.

- Bourne[33] measured spelling mistakes in a bibliographic database. The results showed a misspelling rate that varied between 0.4% and 22.8% (mean 8.8%) for the index terms. The ratio of citations to misspelled terms varied between 0.01% and 0.63% (mean 0.27%).
- Kukich[183] analyzed spelling mistakes in the text messages sent over Bellcore's Telecommunications Network for the Deaf. Of the 10.5% of words automatically flagged as being in error, a manual check found that half were not in the dictionary used and the actual error rate was 5% to 6%
- Kundu and Chaudhuri[185] analyzed 15,162,317 of handwritten Bangla, a highly inflectional and phonetic language spoken in Indian, and 375,723 words of typed (mostly computer input) text. For the handwritten material, a mistake occurred every 122 words (0.82%) and in 53% of cases a word contained a single mistake (25% with two mistakes). The overall rate for typed text was 1.42%, and in 65% of cases a word contained a single mistake (11% with two mistakes).
- Mitton[220] looked at handwritten essays by 925 fifteen year old school children. The rate of mistakes of those classified as *poor* spellers was 63 per 1,000 words; it was 14 per 1,000 for *passable* spellers. The *poor* spellers formed approximately a third of the children and contributed more than half of the mistakes. An analysis of the Wing and Baddeley[325] data found that few mistakes occurred near the start of a word.
- Pollock and Zamora[255] (Table 13) automatically flagged (using a dictionary of 40,000 words) mistakes in 25 million words from a scientific citations database. The overall rate of mistakes was 0.2%, and in 94% a word contained a single mistake (with 60% of mistakes being unique).
- Wing and Baddeley[325] looked at the handwritten exam papers of 40 males applying to be undergraduates at Cambridge University (in engineering, mathematics, or natural sciences). The rate of mistakes was around 1.5% of words written. Of these, 52% were corrected by the subject after they had been made. Mistakes were less likely to be made at the beginning or end of words.
- Yannakoudakis and Fawthrop[328] used information on 1,377 spelling error forms (obtained from published papers and checking written essays) to produce what they referred to as a list of rules

of spelling errors (the mappings from correct one- or two-letter sequences to the incorrect one- or two-letter sequences was seen in their data). They proposed that these spelling errors were based on phonological and letter sequence considerations; for instance, *BATH* may be pronounced *B-A-TH* in Northern England, but *B-AR-TH* or *B-OR-TH* in Southern England, potentially leading to different spelling errors. The letter sequence errors were mostly transpositions, letter doublings, and missed letters.

The words used in some analyses of spelling mistakes are based on words flagged by spell checking programs. These programs usually operate by checking the words in a document against a dictionary of correctly spelled words (letter trigram probabilities have also been used[330]). One problem with the dictionary-based method is that some correctly spelled words will be flagged because they are not in the dictionary, and some incorrectly spelled words will not be flagged because the spelling used happens to match a word in the dictionary. As the number of words in the dictionary increases, the number of correctly spelled words that are flagged will decrease, but the number of unflagged spelling mistakes will also decrease. For instance, the rarely used word *beta* may be a misspelling of the more common *beat*, or its use may be intended. The word *beta* is unlikely to be in a small dictionary, but it will be in a large one. An analysis by Peterson[247] found that the number of possible undetected spelling mistakes increases linearly with the size of the dictionary used (taking no account of word frequency). An analysis by Damerau and Mays[83] found that increasing the size of a spell checker's dictionary from 50,000 to 60,000 words eliminated the flagging of 3,240 correctly spelled words and caused 19 misspellings to be missed (in a 21,910,954 word sample).

identifier
nonword spelling

4.4.3 Nonword spelling

How do people spell nonwords (which may also be dictionary words they are not familiar with)? In the case of spoken languages they have a sequence of sounds on which to base a possible spelling. Studies have found that, for English, people do not always spell a nonword spoken to them using the most common spelling pattern (for the sound sequence heard). The choice of spelling is affected by words they have heard recently. For instance, subjects who heard the word *sweet* just before the nonword */priːt/* tended to spell it as *preet*, while those who heard *treat* first tended to spell it as *preat*. Barry and Seymour[17] have proposed a model based on a set of probabilistic sound-to-spelling mappings (which includes the influence of recently heard words).

identifier
second language spelling

4.4.4 Spelling in a second language

A study by Cook[70] compared the spelling mistakes made by 375 overseas students at the University of Essex, against those made by the students in the Wing and Baddeley[325] study. The results showed that students made fewer omissions (31.5% vs. 43.5%), but more substitutions (31.7% vs. 26.7%), transposition (3.1% vs. 1.4%), and other mistakes. Many of the mistakes were caused by close sound–letter correspondence (e.g., interchanging *a*, *e*, or *i*). There were many more different kinds of changes to consonants by overseas students compared to native speakers (38 vs. 21 different pairs).

A study by Brown[40] compared the spelling abilities of native-English speakers with those for whom it was a second language (47 subjects whose first language varied and included Spanish, French, Japanese, Chinese, German, Hebrew, Arabic). The relative pattern of performance for high/low frequency words with regular/irregular spellings (Table 14) was the same for both native and non-native English speakers.

Table 14: Mean number of spelling mistakes for high/low frequency words with regular/irregular spellings. Adapted from Brown.[40]

	High Frequency Regular Spelling	Low Frequency Regular Spelling	High Frequency Irregular Spelling	Low Frequency Irregular Spelling
Native speaker	0.106	4.213	0.596	7.319
Second language	0.766	7.383	2.426	9.255
Example	cat, paper	fen, yak	of, one	tsetse, ghoul

4.5 Semantic associations

A semantic association occurs in the context of these coding guidelines when source code information causes other information to *pop* into a reader's mind. (It is believed[6] that people's recall of information from long-term memory is based on semantic associations.) Other semantic issues are discussed elsewhere.

[] identifier semantic associations
[] Identifier semantic confusability
[782] semantic priming
[] Identifier semantic usability
[] semantic networks
[] reading kinds of

A particular identifier is likely to have several, out of a large number of possible, attributes associated with it. Which of these attributes a reader will be want to recall can depend on the purpose for reading the code and the kind of reference made in a usage of an identifier. The following are some of these attributes:

- *Identifiers in general.* What it denotes in the programs model of the application (e.g., `Node_Rec` might denote the type of a node record and `total_ticks` might denote a count of the number of clock ticks—its visibility), what other locations in the program's source reference the identifier (locations are usually the names of functions and relative locations in the function currently being read), associated identifiers not present in the source currently visible (e.g., a reference to a structure member may require other members of the same type to be considered), or who has control of its definition (e.g., is it under the reader's control, part of a third-party library, or other members of the project).
- *For an object.* The possible range of values that it might hold (e.g., `total_ticks` might be expected to always contain a positive value and a specification for the maximum possible value may exist), its type. (Some of the reasons for wanting to know this information include the range of possible values it can represent or whether it is a pointer type that needs dereferencing.)
- *For a function call.* A list of objects it references (for any call a subset of this list is likely to be needed, e.g., those objects also referenced in the function currently being read), or the value returned and any arguments required.

The following studies investigated the impact of semantic associations on recall of various kinds of information that might be applicable to developer interaction with identifiers.

- A study by McClelland, Rawles, and Sinclair[214] investigated the effects of search criteria on recall performance after a word-classification task. Subjects were given a booklet with pages containing categories and an associated list of words. For instance, the category *dangerous fish* and the words *shark*, *poison*, *trout*, and *paper*. They were asked to write down how many components (zero, one, or two) each word had in common with the category. After rating all category/word pairs, they then had to write down as many words from each of the lists as possible. The results showed that those words sharing two components with the category (e.g., *shark*) were most frequently recalled, those sharing one component with the category (e.g., *poison*, *trout*) were recalled less often, and those sharing no components (e.g., *paper*) were recalled least often. A second experiment measuring cued recall found the same pattern of performance. A study by Hanley and Morris[132] replicated these results.
- Several studies[202, 266] have investigated readers incidental memory for the location of information in text printed on pages. The results show that for short documents (approximately 12 pages) subjects were able to recall, with reasonable accuracy, the approximate position on a page where information occurred. These memories were incidental in that subjects were not warned before reading the material that they would be tested on location information. Recall performance was significantly lower when the text appeared on a written scroll (i.e., there were no pages).

5 Confusability

identifier confusability

For any pair of letter sequences (an identifier spelling) there is a finite probability that a reader will confuse one of them for the other. Some studies have attempted to measure confusability, while others have attempted to measure similarity. This section discusses the different ways readers have been found to treat two different character sequences as being the same. The discussion of individual issues takes the lead

from the particular study being described in either using the term *confusability* or *similarity*. The studies described here involve carrying out activities such as searching the visual field of view, reading prose, recalling lists, and listening to spoken material. While all these activities are applicable to working with source code, the extent of their usage varies.

The following are the character sequence confusability, or similarity, factors considered to be important in this subsection:

- *Visual similarity.* Letter similarity, character sequence similarity, word shape (looks like)
- *Acoustic confusability.* Word sounds like, similar sounding word sequences
- *Semantic confusability.* Readers' existing knowledge of words, metaphors, categories

A reader of a C identifier may consider it to be a word, a pronounceable nonword, an unpronounceable nonword, or a sequence of any combination of these. This distinction is potentially important because a
nonword effects 782
number of studies have shown that reader performance differs between words and nonwords. Unfortunately many of the published studies use words as their stimulus, so the data on readers' nonword performance is sparse.

distinguishing features

When categorizing a stimulus, people are more likely to ignore a feature than they are to add a missing feature. For instance, *Q* is confused with *O* more often than *O* is confused with *Q*. A study by Plauché, Delogu, and Ohala[252] found asymmetries in subjects' confusion of consonants. For instance, while the spoken consonant */ki/* was sometimes interpreted by their subjects as */ti/*, the reverse did not occur (*/ki/* contains a spectral burst in the 3 to 4 KHz region that is not present in */ti/*).

guideline recommendation enforceable

It is intended that these guideline recommendations be enforceable. This requires some method of measuring confusability. While there are no generally applicable measures of confusability, there is a generally used method of measuring the similarity of two sequences (be they letters, phonemes, syllables, or DNA nucleotides)—the Levenstein distance metric. The basic ideas behind this method of measuring similarity are discussed first, followed by a discussion of the various attributes, which might be sources of confusion.

5.1 Sequence comparison

Levenstein distance

The most common method used to measure the similarity of two sequences is to count the minimum number of operations needed to convert one sequence into the other. This metric is known as the *edit* or *Levenstein* distance. The allowed operations are usually insertion, deletion, and substitution. Some variants only allow insertion and deletion (substitution is effectively one of each), while others include transposition (swapping adjacent elements).

The Levenstein distance, based on letters, of *INDUSTRY* and *INTEREST* is six. One of the possible edit sequences, of length six, is:

```
INDUSTRY delete Y ⇒ INDUSTR
INDUSTR delete R ⇒ INDUST
INDUST substitute D by R ⇒ INRUST
INRUST substitute U by E ⇒ INREST
INREST insert T ⇒  INTREST
INTREST insert E ⇒ INTEREST
```

When all operations are assigned the same weight, the cost of calculating the Levenstein distance is proportional to the product of the lengths, $m \geq n$, of the two sequences. An $O(mn/\log n)$ algorithm is known,[212] but in practice only has lower cost for sequence lengths greater than 262,419. However, in its general form the Levenstein distance can use different weights for every operation. When operations can have different weights, the complexity of the minimization problem becomes $O(mn^2/\log n)$. Possible weighting factors include:

- A substitution may depend on the two items; for instance, characters appearing adjacent on a keyword are more likely to be substituted for each other than those not adjacent.

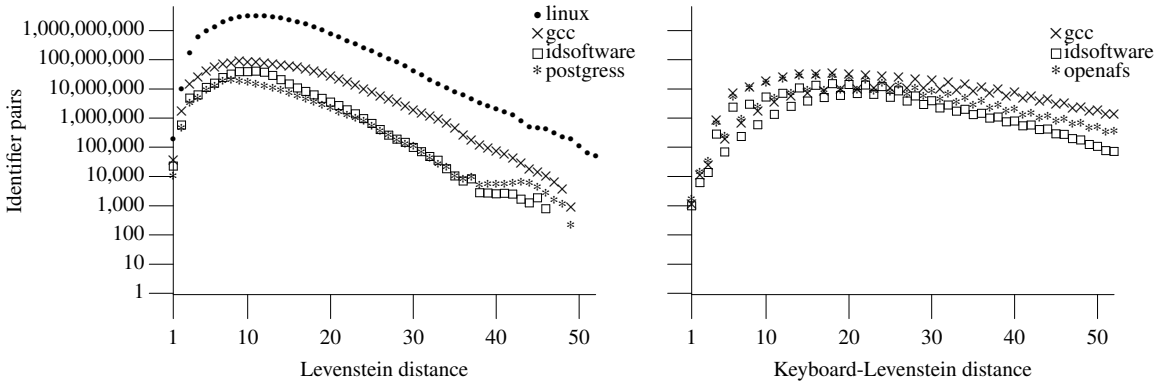


Figure 15: Number of identifiers having a given Levenstein distance from all other identifiers occurring in the visible form of the `.c` files of individual programs (i.e., identifiers in gcc were only compared against other identifier in gcc). The *keyboard-levenstein* distance was calculated using a weight of 1 when comparing characters on immediately adjacent keyboard keys and a weight of 2 for all other cases (the result was normalized to allow comparison against unweighted Levenstein distance values).

- The position of the character within the identifier; that is, a difference in the first character is much more likely to be visually noticed than a difference nearer the end of the identifier
- The visual similarity between adjacent characters; for instance, the Levenstein distance between the identifier `h1tk1` and the two identifiers `__t__` and `kltfh` (based on characters) is the same. However, the identifier `__t__` is much more different visually.

Computing an identifier's Levenstein distance, based on the characters it contains, to every other identifier in a program is potentially a computationally time-consuming operation. Possible techniques for reducing this overhead include the following:

- Reduce the number of identifiers against which the Levenstein distance has to be calculated
- Maintain a dictionary of precompute information for known identifiers. An algorithm by Bunke[45] looks up the nearest neighbor in a dictionary (which needs to hold identifiers converted to some internal form) in linear time, proportional to the length of one of the two strings to be matched (but storage usage is exponential in the length of the dictionary words).
- Map the identifiers into points in a d-dimensional space such that the distances between them is preserved (and less costly to calculate). Jin, Li, and Mehrota[163] describe an algorithm that uses this mapping idea to obtain the set of pairs of identifiers whose distance is less than or equal to k. While this algorithm runs in a time proportional to the total number of identifiers, it is an approximation that does not always find all identifier pairs meeting the distance criteria.
- The possibility of reducing the number of identifier's against which a particular identifier needs to be compared against is discussed elsewhere.

[] identifier guideline significant characters

5.1.1 Language complications

Words often have an internal structure to them, or there are conventions for using multiple words. A similarity measure may need to take a reader's knowledge of internal structure and conventions into account. Three language complications are discussed next —an example of internal word structure (affixes), spelling letter pairs, and a common convention for joining words together (people's names), along with tools for handling such constructs.

agglutinative languages 782

- A word sometimes includes a prefix or suffix. So-called *stemming* algorithms attempt to reduce a word to its common root form. Some algorithms are rule-based, while others are dictionary-based[180] (*creation* can be reduced to *create* but *station* cannot be reduced to *state*). Comparisons of the accuracy of the algorithms has produced mixed results, each having their own advantages.[109, 153] Languages differ in the extent to which they use suffixes to create words. English is at one end of the scale, supporting few suffixes. At the other end of the scale are Hebrew and Slovene (supporting more than 5,000 suffixes); Popovic and Willett[256] found that use of stemming made a significant difference to the performance of an information lookup system.

morpheme 782

- There appears to be patterns to the incorrect characters used in misspelled words. Kernighan, Church, and Gale[169] used the Unix `spell` program to extract 14,742 *misspelled* words from a year's worth of AP wire text (44 million words). These words were used to build four tables of probabilities, using an iterative training process. For instance, $del[x, y]/chars[x, y]$ denotes the number of times the character y is deleted when it follows the character x (in misspelled words) divided by the number of times the character y follows the character x (in all words in the text). The other tables counted insertions, substitutions, and transpositions.
- People's names and source code identifiers often share the characteristic that blocks of characters are deleted or transposed. For instance, "John Smith" is sometimes written as "J. Smith" or "Smith John", and `total_widget` might be written as `tot_widget` or `widget_total`. Searching and matching problems based on different spellings of the same person's name occur in a variety of applications and a number of algorithms have been proposed, including, `LIKEIT`[329] and `Matchsimile`.[232]

5.1.2 Contextual factors

The context in which a letter sequence is read can affect how it is interpreted (and possibly confused with another).

- *Reading sequences of words.* For instance, in the sentence "The child fed the dack at the pond" the nonword *dack* is likely to be read as the word *duck*. Sequences of identifiers separated by other tokens occur in code and are sometimes read sequentially. The extent to which identifier spellings will cause expectations about the spelling of other identifiers is not known.
- *Paging through source code* (e.g., when in an editor). Studies[257] have found that when asked to view rapidly presented lists, subjects tend to mistakenly perceive a nonword as a word that looks like it (the error rate for perceiving words as other words that look like them is significantly smaller).

identifier visual search

- *Searching.* The issues involved in visual search for identifiers are discussed elsewhere.

identifier visual similarity

word visual recognition

5.2 Visual similarity

A large number of factors have been found to influence a reader's performance in visual recognition of character sequences (the special case of the character sequence representing a word is discussed elsewhere). This subsection discusses those visual factors that may cause one character sequence to be confused for another. Individual character similarity is discussed first, followed by character sequence similarity.

identifier introduction

Readers have extensive experience in reading character sequences in at least one natural language. One consequence of this experience is that many character sequences are not visually treated as the sum of their characters. In some cases a character is made more visually prominent by the characters that surround it and in other cases it is visually hidden by these characters. Examples of these two cases include:

- For the first case, the *word superiority effect.* A study by Reicher[262] showed subjects a single letter or a word for a brief period of time. They had been told that they would be required to identify the letter, or a specified letter (e.g., the third), from the word. The results showed that subjects' performance was better when the letter was contained within a word.

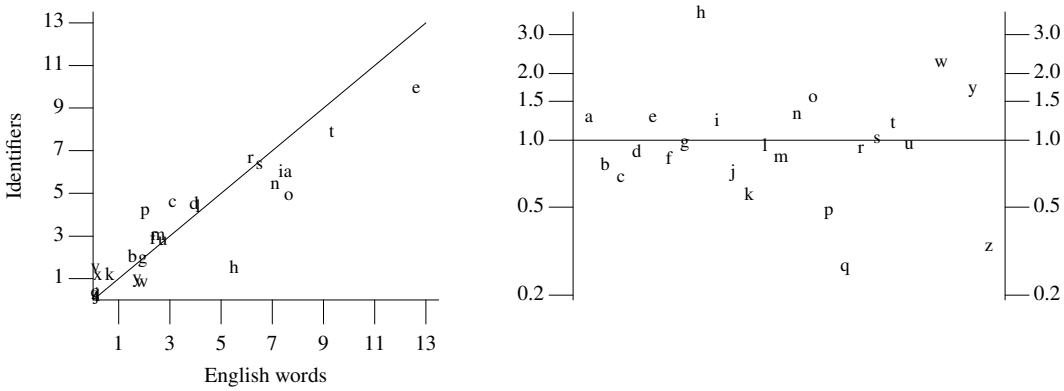


Figure 16: Occurrence of alphabetic letters in English text[285] and identifier names (based on the visible form of the `.c` files; all letters mapped to lowercase). Left graph plots the letter percentage occurrence as (x, y) coordinates and right graph plots the ratio of dividing the English by the identifier letter frequency (i.e., letters above the line are more common in English text than in identifiers; two letters outside the range plotted are $v = 0.0588$ and $x = 0.165$).

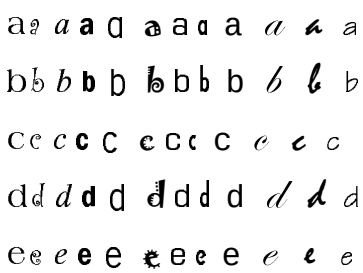


Figure 17: The same glyphs rendered in different fonts.

- For the second case, a study by Holbrook,[145] asked subjects to detect spelling errors in a 700-word story. The spelling errors were created by replacing a letter by either the letter with highest confusability, the letter with the lowest confusability, or one whose confusability was half way between the two extremes. After adjusting for letter frequency, word frequency, and perceived word similarity, the results showed a correlation between measurements of individual letter confusability and subjects' misspelling detection performance.

The following discussion assumes that all of the characters in a particular character sequence are displayed using the same font. Readers' performance has been found[269] to be degraded when some of the characters in a word are displayed in different fonts. However, this issue is not discussed further here.

5.2.1 Single character similarity

character visual similarity

The extent to which two characters have a similar visual appearance is affected by a number of factors, including the orthography of the language, the font used to display them, and the method of viewing them (print vs. screen). Examples of two extremes of similarity (based on commonly used fonts) are the characters `1` (one) and `l` (ell), which are very similar, and the characters `T` and `w` are not visually similar. [782] orthography

In most cases the method used to view source code uses some form of screen. Reading from a printed listing is becoming rare. Even when a printed listing is used, it has usually been produced by a laser printer. The character legibility issues caused by poorly maintained dot matrix or line printers are now almost a thing of the past and are not considered further here.

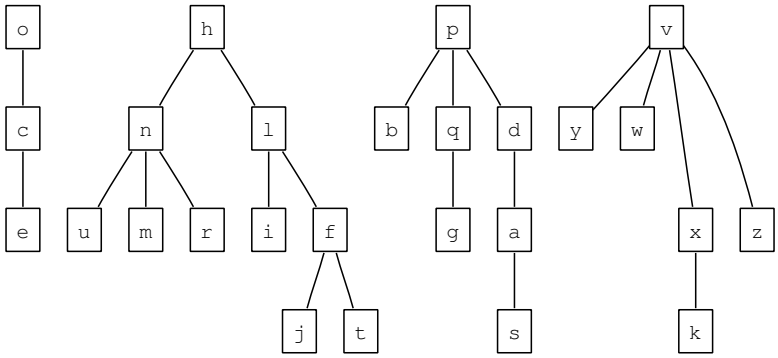


Figure 18: Similarity hierarchy for English letters. Adapted from ###.[?]

Before a character can be recognized, its shape has to be distinguished. The greater the visual similarity between two characters, the greater the probability that one of them will be mistakenly treated as the other. Studies of letter similarity measurement have a long history.[144, 303] These studies have used letters only—there are no statistically significant published results that include digits or common mathematical symbols (as occur in C language source code). The raw data from these measurements is usually a two-dimensional confusion matrix, specifying the degree to which one letter has been estimated (by summing the responses over subjects) to be confusable with another one. In an attempt to isolate the different factors that contribute to this confusion matrix, a multidimensional similarity analysis is sometimes performed.

The following is a discussion of the major published studies. Kuennapas and Janson[182] asked subjects to judge the pairwise similarity of lowercase Swedish letters. A multidimensional similarity analysis on the results yielded nine factors: vertical linearity (e.g., *t*), roundness (e.g., *o*), parallel vertical linearity (e.g., *n*), vertical linearity with dot (e.g., *i*), roundness attached to vertical linearity (e.g., *q*), vertical linearity with crossness (e.g., *k*), roundness attached to a hook (e.g., *å*), angular open upward (e.g., *v*), zigzaggedness (e.g., *z*). Bouma[32] used Dutch subjects to produce confusion matrices for lowercase letters. The letters were viewed at distances of up to 6 meters and at angles of up to 10° from the center of the field of view. The results were found to depend on 16 different factors. Townsend[305] used English subjects to produce confusion matrices for uppercase letters. The letters were briefly visible and in some cases included visual noise or low-light conditions (the aim was to achieve a 50% error rate).

A later study[304] investigated the results from two individuals. Gilmore, Hersh, Caramazza, and Griffin[120] attempted to reduce the statistical uncertainty present in previous studies caused by small sample size. Each English uppercase letter was visually presented on a computer display, and analyzed by subjects a total of 1,200 times. A multidimensional similarity analysis on the published confusion matrices yielded five factors. Gervais, Harvey, and Roberts[118] attempted to fit their confusion matrix data (not published) for uppercase letters to models based on template overlap, geometric features, and spatial frequency content (Fourier transforms). They obtained a correlation of 0.7 between the data and the predictions of a model, based on spatial frequency content.

Boles and Clifford[27] produced a similarity matrix for upper- and lowercase letters viewed on a computer display (many previous studies had used letters typed on cards). A multidimensional similarity analysis on the results yielded three factors—lower- vs. uppercase, curved vs. straight lines, and acute angular vs. vertical.

Given that visual letter recognition is a learned process and that some letters occur much more frequently than others, it is possible that letter confusability is affected by frequency of occurrence. A study by Bouma[32] failed to find a frequency-of-occurrence factor in a letter's confusability with other letters.

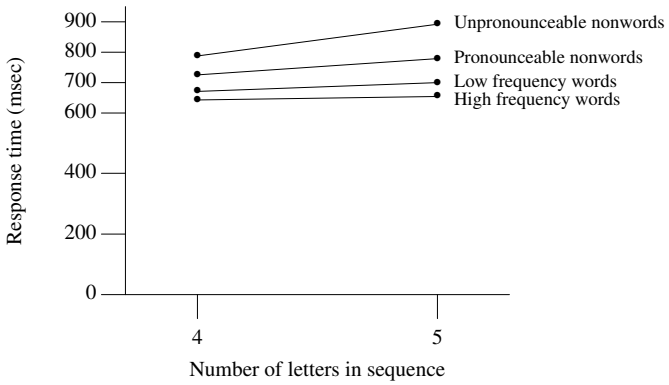


Figure 19: Response time to match two letter sequences as being identical. Adapted from Chambers and Foster.[54]

5.2.2 Character sequence similarity

character sequence similarity
[782] confusability transposed-letter
[782] Levenstein distance

The following is a selection of studies that have investigated readers' performance with sets of character sequences that differ by one or more characters. The study by Andrews suggests that transposition may need to be included in a Levenstein distance calculation.

- A study by Chambers and Foster[54] measured response times in a simultaneous visual matching task using four types of letter sequences—high-frequency words, low-frequency words, pronounceable nonwords, unpronounceable nonwords. Subjects were simultaneously shown two letter sequences and had to specify whether they were the same or different. The results (Figure 19) show the performance advantage for matching words and pronounceable nonwords. Measurements were also made when the letter sequences differed at different positions within the sequence; Table 15 shows the response times for various differences.

Table 15: Response time (in milliseconds) to fail to match two letter sequences. Right column is average response time to match identical letter sequences. Columns are ordered by which letter differed between letter sequences. Adapted from Chambers and Foster.[54]

	All Letters	First Letter	Third Letter	Fifth Letter	Same Response
Words	677	748	815	851	747
Pronounceable nonwords	673	727	844	886	873
Unpronounceable nonwords	686	791	1,007	1,041	1,007

Chambers and Foster explained the results in terms of three levels of letter sequence identification: the word level, the letter cluster level, and the letter level. The higher the level at which operations are performed, the fewer are required for a fixed number of letters. The increasing response time as the differing letter moves further to the right of the word suggests a left-to-right order of processing. However, performance when all letters differ is much better than when only the first letter differs. This behavior would not occur if a strictly left-to-right comparison was performed and suggests some *whole word* processing occurs—also see.[138]

- A study by Eichelman[95] measured response times in a simultaneous visual matching task where letter sequences (either words or randomly selected) varied in the number letters that differed or in case (lower vs. upper). The results (Figure 20) show how response time increases with the number of letters in a sequence and decreases as the number of letters that are different increases.

  In a second simultaneous visual matching task some letter sequences were all in uppercase, while others were all in lowercase. Subjects were told to compare sequences, ignoring letter case. The results

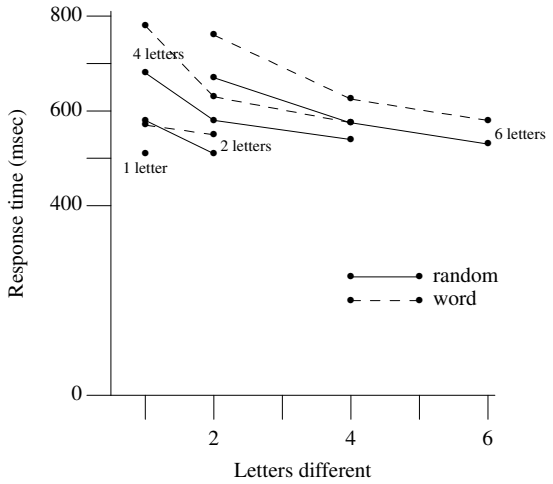


Figure 20: Time taken (in milliseconds) to match a pair of letter sequences as being identical—for different number of letters in the sequence and number of positions in the sequence containing a nonmatching letter. Adapted from Eichelman.[95]

showed that when the case used in the two letter sequences differed, the time taken to match them as being identical increased for both words and random selection. This pattern of response is consistent with subjects performing a visual match rather than recognizing and comparing whole words. For this behavior, a change of case would not be expected to affect performance when matching words.

word reading individual

- Neighborhood frequency effects. A study by Grainger, O'Regan, Jacobs, and Segui[126] found an interaction (response time and error rate) between the initial eye fixation point within a word and the position of the letter differentiating that word from another (this behavior is only possible if readers have a preexisting knowledge of possible spellings they are likely to encounter, which they will have for words).

Mr. Chips

illusory conjunctions 782

- Perceptual interaction between two adjacent, briefly presented, words (as might occur for two operands in an expression when paging through source in an editor). For instance, subjects have reported the adjacent words *line* and *lace* as being *lane* or *lice*.[312]

confusability transposed-letter

- Transposed-letter confusability. A study by Andrews[7] found that so-called *transposed letter pair* words (e.g., *salt-slat*) affected subjects' performance in some cases (low-frequency words; nonwords were not affected). A model of internal human word representation based on letters and their approximate position within a word was discussed.
- A study by Ziegler, Rey, and Jacobs[334] found that speed of recognition of words whose letters were not readily legible was proportional to the log of their frequency of occurrence and approximately linear on what they defined as letter confusability. The error rate for correctly identifying words was proportional to the number of orthographic neighbors of a word, and a measure of its orthographic redundancy.

word shape

5.2.2.1 Word shape

The term *whole word shape* refers to the complete visual appearance of the sequence of letters used to form a word. Some letters have features that stand out above (ascenders—*f* and *t*) and below (descenders —*q* and *p*) the other letters, or are written in uppercase. Source code identifiers may also have another shape-defining character—the underscore _. Words consisting of only uppercase letters are generally considered to have the same shape. The extent to which *whole word shape* affects visual word-recognition performance is still being debated (see Perea and Rosa[245] for a recent discussion). In the following two studies a whole word shape effect is found in one case and not in the other.

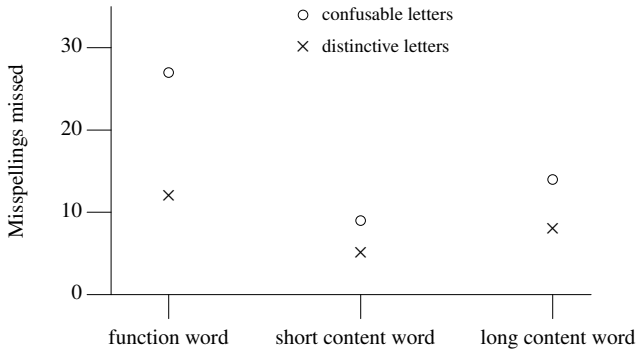


Figure 21: Percentage of misspellings not detected for various kinds of word. Adapted from Paap, Newsome, and Noel.[239]

- A study by Monk and Hulme[222] asked subjects to quickly read a paragraph. As a subsidiary task they were asked to circle spelling errors in the text. The spelling errors were created by deleting or substituting a letter from some words (in some cases changing the shape of the word). The results (Table 16) showed that when lowercase letters were used, a greater number of misspelled words were circled when the deletion also changed the shape of the word. When a mixture of letter cases (50% lowercase/uppercase) was used, differences in word shape were not sufficient to affect misspelling detection rates (this study has been replicated and extended[137]).

Table 16: Proportion of spelling errors detected (after arcsin transform was applied to the results). Adapted from Monk and Hulme.[222]

	Same Lowercase Word Shape	Different Lowercase Word Shape	Same Mixedcase Word Shape	Different Mixedcase Word Shape
Letter deleted	0.554	0.615	0.529	0.517
Letter substituted	0.759	0.818	0.678	0.680

- A study by Paap, Newsome, and Noel[239] modified an earlier study by Haber and Schindler,[131] which had found a word shape effect. Subjects were asked to read a passage at their normal reading speed, circling any misspelled words as they read. They were told that they would be tested for comprehension at the end. Four kinds of misspellings were created. For instance, the letter *h* in *thought* was replaced by: (1) *b* (maintain shape and confusable letter), (2) *d* (maintain shape and distinct letter), (3) *n* (alter shape and confusable letter), or (4) *m* (alter shape and distinct letter). The results (Figure 21) showed that many more misspellings went undetected when the letters were confusable (irrespective of word shape) than when they were distinctive (irrespective of word shape).

5.3 Acoustic confusability

acoustic confusability

C source code is not intended to represent a spoken language. However, it is always possible to convert an identifier's character sequence to some spoken form. This spoken form may be a list of the individual characters or readers may map one or more characters (graphemes) to sounds (phonemes). This subsection does not concern itself with the pronunciation used (the issue of possible pronunciations a developer might use is discussed elsewhere), but discusses the various possibilities for confusion between the spoken form of different identifiers.

[] standard specifies form and interpretation

[782] grapheme
[782] phoneme

[] word pronounceability

The studies discussed in this subsection used either native British or American speakers of English. It is recognized that the pattern of sounds used by different languages varies. The extent to which the results,

relating to sound similarity, obtained by the studies discussed here are applicable to speakers of other languages is not known. While many of the studies on acoustic similarity use spoken material as input to subjects, not written material, most of the studies described here used written material.

A number of studies have found what has become known as the *dissimilar immunity effect*.[14] The effect is that the recall of phonologically dissimilar items on a list is not affected by the presence of similar items (i.e., recall is the same as if the items appeared in a completely dissimilar list). A recent study[97] found evidence that similar items enhanced memory for the order of dissimilar items between two similar items.

The first subsection discusses the issue of measuring the degree to which the pronunciations of two identifiers sound alike. This is followed by subsection discussing studies of letter acoustic confusion and memory performance for lists of words that sound alike.

5.3.1 Studies of acoustic confusion

A number of studies have investigated the acoustic confusion of letters and digits. Two of these (both using British subjects) published a confusion matrix. In the study by Conrad[66] (whose paper only contained a data on a subset of the letters—see Morgan[224] for data on all letters), subjects listened to a recording of spoken letters into which some noise had been added and were asked to write down the letters they heard. The study by Hull[152] included letters and digits.

In a study by Morgan, Chambers, and Morton[225] subjects listened to a list of digits (spoken by either an American or Australian female or an English male) and after hearing the list were asked to write down the digits. A total of 558 subjects listened to and recalled 48,402 digit lists. Confusion matrices for the different speakers and the recognition and memory tasks were published.

What are the factors that affect the confusability of two letters? Both Morgan[224] and Shaw[277] performed multidimensional scaling analysis[12] on the Conrad and Hull data. Morgan presented an analysis using $n = 3$, but suggested that larger values may be applicable. Shaw treated the letters phonologically, using $n = 5$ isolated phonological factors for three of the dimensions (open vs. closed vowels, vowel+consonant sound vs. consonant+vowel sound, and voiced vs. unvoiced consonants). There good correlation between the ordering of data points in one of the dimensions (open vs. closed vowels) and the first formant frequency of English vowels (see Cruttenden, tables 3, 4, and 5[78]). This correlation was also found by Morgan, Chambers, and Morton[225] in their study of digits.

While these results offer a possible explanation for the factors affecting letter confusion and data points from which calculations can be made, it is important to bare in mind that the pronunciation of letters will depend on a person's accent. Also, as the measurements in Cruttenden[78] show, the relative frequencies of formants (assuming this is a factor in confusability) differ between males and females.

5.3.1.1 Measuring sounds like

coding guidelines introduction

One of the design aims of guideline recommendations is to support automatic enforcement. In order to measure the acoustic similarity of two identifiers, it is necessary to be able to accurately convert their spellings to sound and to have a method of measuring the similarity of these two sounds. There are a number of issues that measurements of sounds like need to address, including:

You can lead a horse to water, but a pencil must be lead. Wind the clock when the wind blows.

- Developers are well aware that source code identifiers may not contain natural language words. It is likely that an identifier which is a nonword will be pronounced using the grapheme-to-phoneme route. For instance, the pronunciation of the nonword `gint` rhyming with the word `hint` rather than `pint`. The study by Monsell et al.
- Some identifiers contain more than one word. Readers recognize a word as a unit of sound. As such, two identifiers containing the same two words would have the same sound if those words occurred in the same order. For instance, `count_num` and `num_count` could be said to have a Levenstein distance of one based on the word as the edit unit.

characters mapping to sound

The issue of converting character sequences to sounds is discussed elsewhere. A number of studies have

[12] Multidimensional scaling is a technique that attempts to place each result value in an n-dimensional space (various heuristics are

attempted to measure the sound similarity of two words, the following are three of them:

- Frisch[104] describes a method of computing a similarity metric for phonological segments (this is used to calculate a relative degree of confusability of two segments, which is then compared against English speech error data). [782] syllable
- Mueller, Seymour, Kieras and Meyer[228] defined a phonological similarity metric between two words based on a multidimensional vector of psychologically relevant (they claimed) aspects of dissimilarity. The dimensions along which the metric was calculated included the extent to which the words rhyme, the similarity of their stress patterns, and the degree to which their syllable onsets match. [782] syllable
- Lutz and Greene[205] describe a system that attempts to predict probable pronunciations of personal names based on language-specific orthographic rule sets. This information was then used to automatically measure the phonological similarity between the name and potential matches in a database.

5.3.2 Letter sequences

A study by Conrad[67] visually presented subjects with six consonants, one at a time. After seeing all the letters, subjects had to write down the recalled letters in order. An analysis of errors involving letter pairs (Table 17) found that acoustic similarity (AS) was a significant factor.

Table 17: Classification of recall errors for acoustically similar (AS), acoustically dissimilar (AD) pairs of letters. Semi-transpose refers to the case where, for instance, *PB* is presented and *BV* is recalled (where *V* does not appear in the list). Other refers to the case where pairs are both replaced by completely different letters. Adapted from Conrad.[67]

Number Intervening Letters	Transpose (AS)	Semi-transpose (AS)	Other (AS)	Transpose (AD)	Semi-transpose (AD)	Other (AD)	Total
0	797	446	130	157	252	207	1,989
1	140	112	34	13	33	76	408
2	31	23	16	2	18	56	146
3	12	20	12	1	5	23	73
4	0	4	1	0	2	7	14
Total	890	605	193	173	310	369	2,630

5.3.3 Word sequences

As the following studies show, people recall fewer words from a list containing similar sounding words than dissimilar sounding words. The feature-based model of immediate memory[230] explains the loss of information from short-term memory in terms of interference between the items being remembered rather than their memory traces decaying over time. This model predicts that similar sounding words will interfere with each other more than dissimilar sounding words (it also accounts for the recency effect and temporal grouping). A subsequent enhancement to the model[233] enabled it to account for the word-length effect (memory performance is worse for items that take longer to pronounce).

- A study by Baddeley[12] dramatically showed the effects of acoustic similarity on recall performance from short-term memory. Subjects were visually presented with either a list of acoustically similar words (e.g., *man*, *cab*, *can*, *cad*, *cap*, *mad*, *map*, etc.), or a list of words that were not acoustically similar (e.g., *few*, *pit*, *cow*, *pen*, *bar*, *hot*, *bun*, etc.). After a delay of zero, four, or eight seconds, during which they had to write down digits that were slowly read to them, they had to write down the presented word list. The results (Figure 22) show the significant impact acoustic similarity can have on recall performance.

used to select n, with each dimension being interpreted as a factor contributing to the final value of the result obtained).

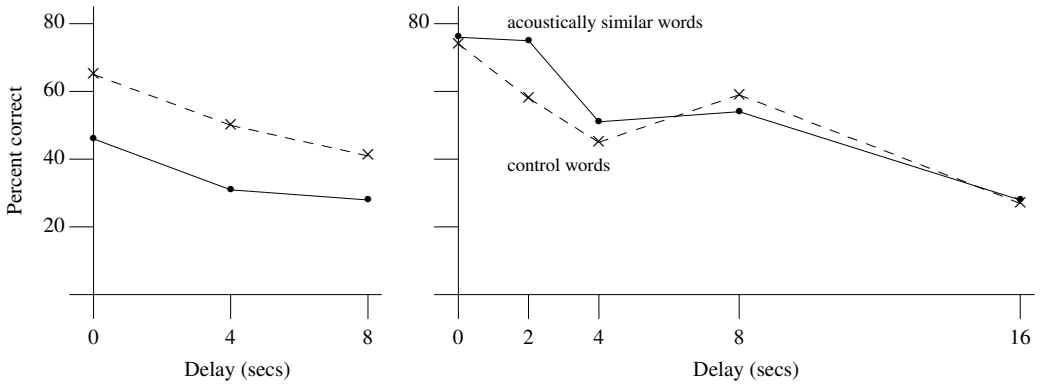


Figure 22: Left graph is the rate of forgetting of visually presented lists of four words containing the same (solid line) or different (dashed line) vowels. Right graph is the rate for two lists, one containing three acoustically similar words (solid line) and the other five control words (dashed line). Adapted from Baddeley.[12]

- A study by Daneman and Stainton[85] asked subjects to carefully read a passage (subjects were given a comprehension test) and proofread it for spelling errors. The spelling errors were either homophones (e.g., *meet* replaced by *meat*) or not (e.g., *meet* replaced by *meek*). Homophone misspellings were less likely to be detected than the non-homophone ones. A study by Van Orden[314] asked subjects to make a semantic category decision on a visually presented word. For instance, they were told "answer yes/no if the word is a flower" and presented with either *ROSE*, *ROWS*, or *ROBS*. The results showed that subjects were significantly more likely to answer yes to words that were homophones of words that were members of the category.

malapropisms

- People do not always say the word they intended to speak. Two well-known kinds of mistake are: (1) *malapropisms*, using a word that has a similar sound but a different meaning ("the geometry of contagious countries"), and (2) *spoonerisms*, the transposition of syllables between two words (*blushing crow* instead of *crushing blow*). A study by Fay and Cutler[98] located 183 malapropisms in a collection of more than 2,000 speech errors. They found that in 99% of cases the target and erroneous word were in the same grammatical category, in 87% of cases they contained the same number of syllables, and in 98% of cases they shared the same stress pattern.

Spoonerisms 782

- *silent* letters. Early studies[73] suggested that silent letters (i.e., not pronounced) in a word were more likely to be go unnoticed (i.e., their misspelling not detected in a proofreading task) than letters that were pronounced. Later studies[92] showed that the greater number of search errors occurred for high-frequency function words (e.g., *the*), not for content words. The search errors were occurring because of the high-frequency and semantic role played by the word, not because letters were either silent or pronounced.

- A study by Coltheart[64] found that it did not matter whether the same, visually presented, words were used on each memory trial, or different words were used every time; recall of similar sounding words was significantly lower (62–69% vs. 83–95% when recall in the correct order was required, and 77% vs. 85–96% when order-independent recall was required).

Identifier semantic confusability

5.4 Semantic confusability

Developers often intend identifier names to convey semantic information about what an identifier represents. Because of the small number of characters that are usually used, the amount of semantic information that can be explicitly specified in the name is severely limited. For this reason developers make assumptions (usually implicitly) about knowledge they share with subsequent readers. Semantic confusion occurs when

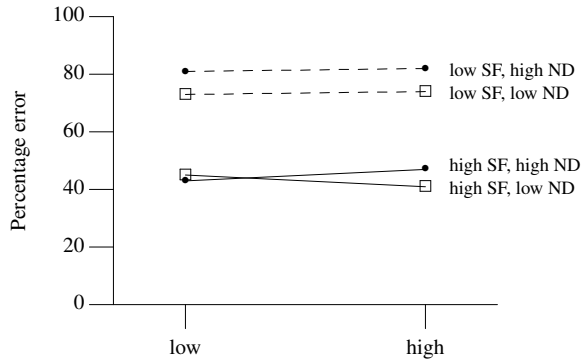


Figure 23: Error rate at low and high neighborhood frequency. Stimulus (drug name) frequency (SF), neighborhood density (ND). Adapted from Lambert, Chang, and Gupta.[190]

a reader of the source does not meet the shared knowledge assumptions made by the creator of the identifier name.

This coding guideline section does not make any recommendations relating to semantic confusability. Given the extent to which people implicitly treat their own culture as *natural*, it is difficult to see how even detailed code reviews can reliably be expected to highlight culturally specific identifier naming assumptions made by project team members. However, the following subsections attempt to give a flavor of some of the possible confusions that might occur; the issues covered include natural language, metaphor, and category formation.

[] code reviews

5.4.1 Language

language affecting thought

Natural language issues such as word order, the use of suffixes and prefixes, and ways of expressing relationships, are discussed elsewhere.

[] human language characteristics

In written prose, use of a word that has more than one possible meaning, *polysemy*, can usually be disambiguated by information provided by its surrounding context. The contexts in which an identifier, containing a polysemous word, occur may not provide enough information to disambiguate the intended meaning. The discussion on English prepositions provides some examples.

[] Prepositions

5.4.1.1 Word neighborhood

Word neighborhood effects have been found in a number of contexts. They occur because of similarities between words that are familiar to a reader. The amount of familiarity needed with a word before it causes a neighborhood effect is not known. Studies have found that some specialist words used by people working in various fields show frequency effects. The study described next shows a word neighborhood effect—the incorrect identification of drug names.[13]

[782] neighborhood identifier

[782] words domain knowledge

A study by Lambert, Chang, and Gupta[190] investigated drug name confusion errors. Drug names and their U.S. prescription rates were used, the assumption being that prescription rate is a good measure of the number of times subjects (forty-five licensed, practicing pharmacists) have encountered the drug name. Subjects saw a drug name for three seconds and were then asked to identify the name. The image containing each name had been degraded to a level comparable to that of a typewritten name received through a fax machine with a dirty print cartridge that was running out of ink.

The results found that each subject incorrectly identified 97 of 160 drug names (the degraded image of the drug name being responsible for the very high error rate of 60.6%). Both the frequency of occurrence of drug names and their neighborhood density were found to be significant factors in the error rate (Figure 23). Neighborhood frequency was not a significant factor.

[782] neighborhood identifier

[13] Errors involving medication kill one person every day in the U.S., and injure more than one million every year; confusion between drug names that look and sound alike account for 15% to 25% of reported medication errors.

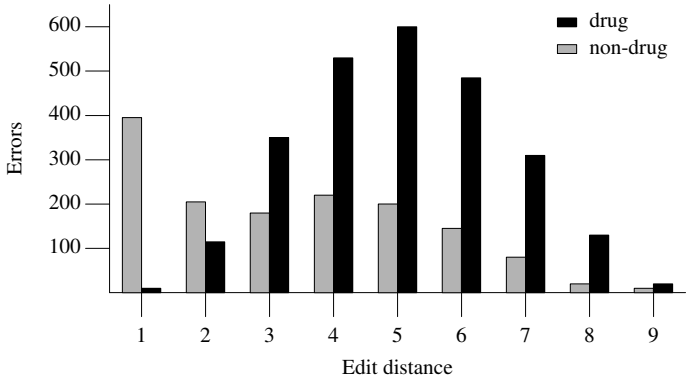


Figure 24: Number of substitution errors having a given edit distance from the correct response. Grey bars denote non-drug-name responses, while black bars denote responses for known drug names. Based on Lambert, Chang, and Gupta.[190]

An analysis of the kinds of errors made found that 234 were omission errors and 4,128 were substitution errors. In the case of the substitution errors, 63.5% were names of other drugs (e.g., *Indocin*® instead of *Indomed*®), with the remaining substitution errors being spelling-related or other non-drug responses (e.g., *Catapress* instead of *Catapres*®). Figure 24 shows the number of substitution errors having a given edit distance from the correct response.

All character sequences ever encountered by a reader can potentially have a word-frequency effect. The character sequences most familiar to developers are those of their native language.

identifier usability

6 Usability

Identifier usability is in the eye (and life experienced mind) of the beholder. The original author of the source, subsequent maintainers of that source, and the managers responsible for products built from the source are likely to have different reasons for reading the source and resources (time and past experience) available to them. Cognitive effort minimization and speed/accuracy trade-offs are assumed to play an important role in identifier usability. As well as discussing visual, acoustic, and semantic factors, this subsection also covers the use of cognitive resources, initial implementation versus maintenance costs (an issue common to all coding guideline), and typing.

identifier developer interaction
cognitive effort
effort vs. accuracy
decision making

coding guidelines importance

Management may show an interest in the spelling used for identifiers for a number of reasons. These reasons, which are not discussed further here, include:

- Some vendors provide interfaces via callable functions or accessible objects, making the names of identifiers visible to potentially millions of developers. Or, a company may maintain internal libraries that are used across many development projects (a likely visibility in the hundreds or perhaps low thousands of developers, rather than millions). In this case customer relation issues are often the most significant factor in how identifier spellings are chosen.
- Project control (or at least the appearance of), often involves the creation of identifier naming guideline documents, which are discussed elsewhere. Other forms of project control are code reviews. These reviews can affect identifier naming in code whether it is reviewed or not, people often use a different decision strategy when they know others will be evaluating their choice.

identifier other guideline documents
code reviews

justifying decisions

6.1 C language considerations

scope

For the majority of identifiers, scope is the most important attribute in deciding their usage patterns:

- *local scope.* This is the most common form of identifier usage, both in terms of number of identifiers defined (Table) and number of occurrences in the visible source of function definitions. Individual

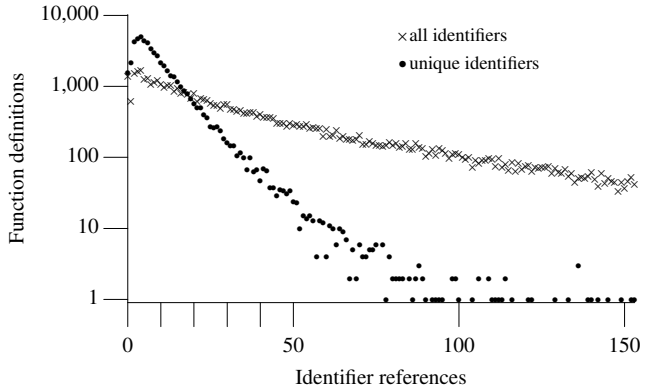


Figure 25: Number of identifiers referenced within individual function definitions. Based on the translated form of the books benchmark programs.

identifier usage is restricted to a relatively small subset of the source code. It is possible for local identifiers in different parts of the source code to share the same name, yet refer to different entities. Identifiers in local scope can be used and then forgotten about. Individually these identifiers may only be seen by a small number of developers, compared to those at global scope.

- *global scope.* While use of this kind of identifier may be restricted to a single source file, it often extends to the entire source code of a program. While, on average, they may be individually referenced more often than individual local identifiers, these references tend to be more widely scattered throughout the source code. Although it may be possible to use and then forget about some identifiers at global scope, it is much more likely that they will need to be recalled, or recognized, across a wider range of program source. They are also much more likely to be used by all the developers working on a project.
- *header file contents.* Over time developers are likely to learn the names of identifiers that appear in included headers. Only identifiers appearing in the most frequently read functions will be familiar. In the early stages of learning about a function, developers are likely to think of identifiers in terms of their spelling; they have yet to make automate the jump to their semantic meaning. How does this impact a reader's analysis of expressions and sequences of statements? It is likely that while being analyzed their component parts will be held in short-term memory.

The majority of references to identifiers occur within expressions and most declarations declare identifiers to be objects (Table).

[] expressions

6.2 Use of cognitive resources

identifier
cognitive resource usage
[] attention
[] cognitive effort
[] developer differences

The cognitive resources available to developers are limited. An important aspect of usability is making the optimum use of the available cognitive resources of a reader. Overloading the available resources can lead to a significant increase in the number of errors made.

Different people have different amounts of cognitive resources at their disposal. A general discussion of this issue is given elsewhere. This subsection discusses people's cognitive performance characteristics from the perspective of processing character sequences. While recognizing that differences exist, it does not discuss them further.

A study by Hunt, Lunneborg, and Lewis[156] investigated various performance differences between subjects classified as either *low verbal* or *high verbal*. The Washington Pre-College Test was used to measure verbal ability (a test similar to the Scholastic Achievement Test, SAT). In a number of experiments there was a significant performance difference between low and high verbal subjects. In some cases, differences

in performance were only significant when the tests involved existing well-learned patterns. For instance, high verbals performed significantly better than low verbals in remembering written sequences of syllables when the sequences followed English usage patterns; the performance difference was not very large when nonsense syllable sequences were used.

identifier resource minimization

6.2.1 Resource minimization

Minimizing the amount of resources needed to accomplish some goal is an implicit, if not explicit, human aim in most situations. Studies have found that individuals and groups of people often minimize their use of resources implicitly, without conscious effort.

automatization

In the context of this coding guideline resource minimization is a complex issue. The ways in which developers interact with identifiers can vary, as can the abilities (resources available) to individual developers, and management requirements target particular developers (e.g., having certain levels of experience with the source, or having particular cultural backgrounds).

identifier developer interaction developer differences

Zipf's law

Zipf noticed a relationship between the frequency of occurrence of some construct, created by some operation performed by people, and the effort needed to perform them. He proposed an explanation based on the principle of least effort. What has become known as Zipf's law[335] states a relationship between the rank and frequency of occurrence of some construct or behavior. Perhaps its most famous instantiation relates to words, $r = C/f_r$—where r is 1 for the most frequently occurring word, 2 for the second most frequently occurring, and so on; f_r is the number of times the word of rank r occurs; and C is a constant. According to this law, the second most common word occurs half as many times as the most commonly occurring word (in English this is *the*), the third most common occurs $2/3$ times as often as the second most common, and so on.

Zipf's law has been found to provide a good approximation to many situations involving a cost/effort trade-off among different items that occur with varying degrees of frequency. Further empirical studies[258] of word usage and theoretical analyses have refined and extended Zipf's original formulation.

However, while it is possible to deduce an inverse power law relationship between frequency and rank (Zipf's law) from the principle of least effort, it cannot be assumed that any distribution following this law is driven by this principle. An analysis by Li[196] showed that words in randomly generated texts (each letter, including the space character, being randomly selected) exhibit a Zipf's law like frequency distribution.

The relatively limited selection pressure on identifier spelling (the continued existence of source code does not depend on the spelling of the identifiers it contains and developers are rarely judged by the identifier spelling they create) does not necessarily mean that identifier spellings don't evolve. A study by Kirby[172] found that languages can evolve simply through the dynamics of learning, no selection of learners (i.e., killing off those that fail to reach some minimum fitness criteria) is needed.

A plot of the rank against frequency of identifier spellings (Figure 26) shows a good approximation to a straight line (the result expected from measuring behavior following Zipf's law). Your author cannot provide a reasonable explanation for this behavior.[14]

identifier information extraction

6.2.2 Rate of information extraction

Reading a character sequence requires identifying each of the characters. The results of a study by Miller et al. found that the amount of information, about a nonword, recalled by subjects was approximately proportional to the time interval it was visible to them. However, the unit of perception used by subjects was not the character. Subjects made use of their knowledge of native language character ordering relationships to chunk sequences of characters into larger perceptual units. A number of studies have suggested various candidates (e.g., syllables, graphemes) for the perceptual reading unit, and these are also described here.

developer computational power

A study by Miller, Bruner, and Postman[219] measured the amount of information subjects remembered about a briefly presented nonword. The nonwords were constructed so as to have different orders of approximation to existing English words (Table 18). Subjects saw a single nonword for a duration of 10, 20, 40,

[14] An explanation for why a ranking of cities by their population follows Zipf's law has been provided by Gabaix,[113] who showed it to be a statistical consequence of individual city population growth following Gibrat's law.

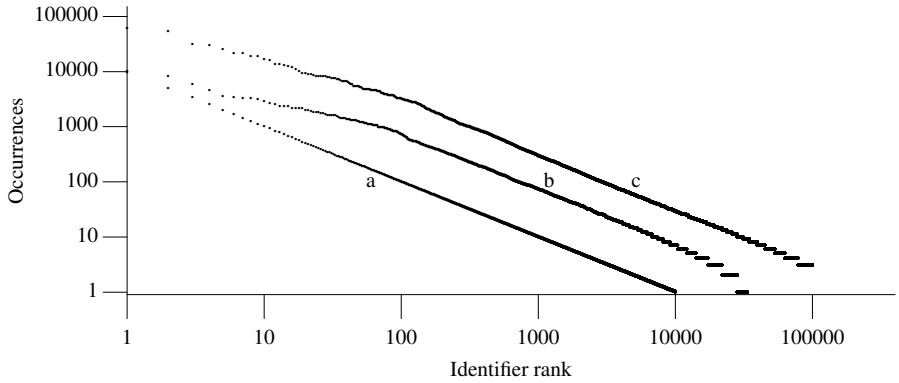


Figure 26: Frequency of identifiers having a particular spelling plotted against the number of occurrences (the content of comments has been excluded) in the visible form of (b) Mozilla, and (c) Linux 2.4 kernel; (a) is a distribution following Zipf's law with the most common item occurring 10,000 times. A dot is appears for each identifier rank.

100, 200, or 500ms. They then had to write down the letters seen and their position in the nonword (using an answer sheet containing eight blank squares).

Table 18: Examples of nonwords. The 0-order words were created by randomly selecting a sequence of equally probable letters, the 1-order words by weighting the random selection according to the probability of letters found in English words, the 2-order words by weighting the random selection according to the probability of a particular letter following the previous letter in the nonword (for English words), and so on. Adapted from Miller[219]).

0-order	1-order	2-order	4-order
YRULPZOC	STANUGOP	WALLYLOF	RICANING
OZHGPMTJ	VTYEHULO	RGERARES	VERNALIT
DLEGQMNW	EINOAASE	CHEVADNE	MOSSIANT
GFUJXZAQ	IYDEWAKN	NERMBLIM	POKERSON
WXPAUJVB	RPITCQET	ONESTEVA	ONETICUL
VQWVBIFX	OMNTOHCH	ACOSUNST	ATEDITOL
CVGJCDHM	DNEHHSNO	SERRRTHE	APHYSTER
MFRSIWZE	RSEMPOIN	ROCEDERT	TERVALLE

The results (Figure 27) show a consistent difference among the order of approximation for all presentation times. Miller et al. proposed that subjects had a fixed rate of information intake. The performance difference was caused because the higher-order letter sequences had a lower information content for the English language speaking subjects. Had the subjects not known English speakers, but Japanese speakers for instance, they would have had no knowledge of English letter frequency and the higher-order letter sequences would have contained just as much information as the lower-order ones.

This study (reproduced by Baddeley[13] using spoken rather than visual presentation of letters) shows that developers will need more time to process identifier spellings having a character sequence frequency distribution that does not follow that of their native language. In those cases where source is quickly scanned, a greater number of characters in a sequence (and their positions) are available for recall if they have frequency distribution of the readers' native language.

If individual characters are not the unit of perception and recall used by readers when processing words, what is? The following are a number of proposals:

- A study by Spoehr and Smith[287] asked subjects to identify some of the letters in a briefly presented letter sequence (the subjects did not know which letters until after they had seen the letter sequence). The results showed that subjects' perceptual accuracy for a word is correlated with the number of re-

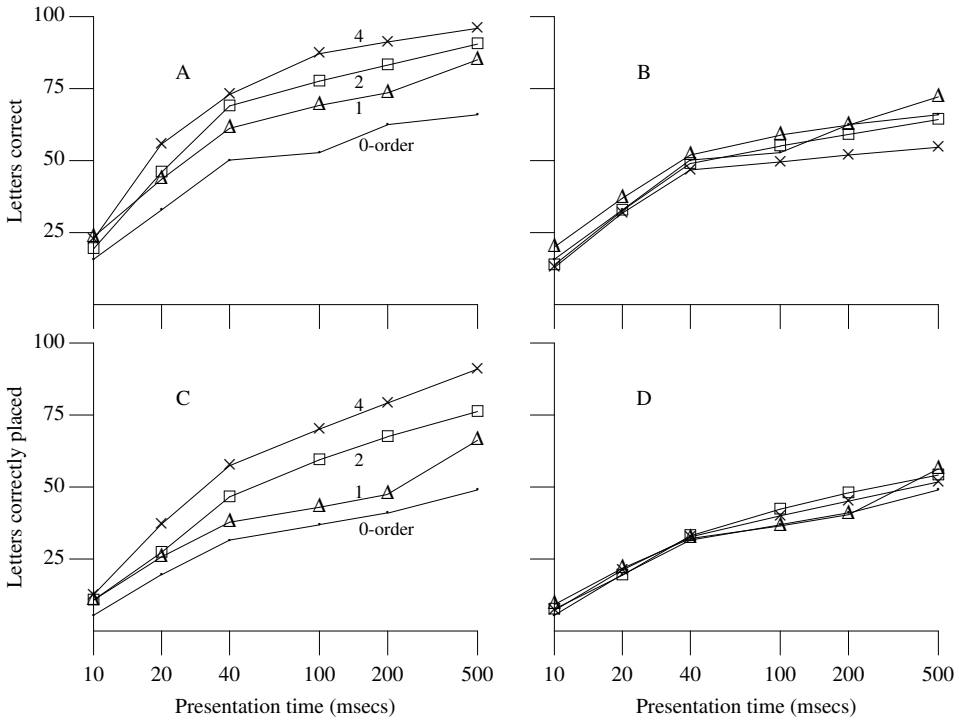


Figure 27: Number of correct letters regardless of position (A), and number of correct letters placed in the correct position (C). Normalizing for information content, the corresponding results are (B) and (D), respectively. Plotted lines denote 0-, 1-, 2-, and 4-order approximations to English words (see Table 18). Adapted from Miller, Bruner, and Postman.[219]

coding steps needed to convert it into speech. For instance, the letter sequences *LSTB*, *BLST*, *BLOST*, and *BLAST* are a sequence not matching English rules, matching English rules but omitting a vowel, a pronounceable nonword, and a word, respectively. They are reproduced correctly in 66%, 70%, 78%, and 82% of cases, respectively. The results are consistent with the letters of a word first being parsed into syllables.

- A study by Rey, Ziegler, and Jacobs[264] asked subjects (English and French, using their respective native language) to specify whether a letter was contained within a briefly displayed word. The results found that the response time was longer when the searched for letter was contained within a multi-letter grapheme (phoneme similarity was taken into account). These results are consistent with graphemes being the perceptual reading unit.

grapheme 782

6.2.3 Wordlikeness

identifier
wordlikeness

The extent to which a nonword is similar to words belonging to a particular natural language is called its *wordlikeness*. Wordlikeness is a cognitive resource in that it represents a person's accumulated reading experience. Wordlikeness has a strong influence on how quickly and accurately nonwords are processed in a variety of tasks.

How is *wordlikeness* measured? Coleman and Pierrehumbert[61] trained a general phonological word grammar on a list of English words whose phonology was known. The result was a grammar whose transition probabilities correspond to those of English words. This grammar was used to generate a list of letter sequences and subjects were asked to judge their *wordlikeness*. The results showed that the log probability of the phonological rules used in the generation of the complete letter sequence had a high correlation with subjects' judgment of wordlikeness. A study by Frisch, Large, and Pisoni[105] replicated and extended these results.

A number of studies have found differences in people's performance in tasks involving words or nonwords, including:

word nonword effects

- *Visual comparison of words and nonwords.* Studies of performance in comparing two-letter sequences have found that response time is faster when the letter sequence represents a word rather than a nonword. [] character sequence similarity
- *Naming latency.* A study by Weekes[322] found that naming latency for high-frequency words was not significantly affected by the number of letters (between three and six), had some affect for low-frequency words, and had a significant affect for nonwords (this study differed from earlier ones in ensuring that number of phonemes, neighborhood size, bigram frequencies, and other linguistic factors were kept the same). The error rate was not found to vary with number of letters.
- *Short-term memory span.* A study by Hulme, Maughan, and Brown[154] found that subjects could hold more words in short-term memory than nonwords. Fitting a straight line to the results, they obtained:

$$wordspan = 2.4 + 2.05 * speechrate \quad (1)$$

$$nonwordspan = 0.7 + 2.27 * speechrate \quad (2)$$

They concluded that information held in long-term memory made a contribution to the short-term memory span of that information.

Wordlikeness may involve more than using character sequences that have an n-th order approximation to the target language. For instance, readers of English have been found to be sensitive to the positional frequency of letters within a word.[213] Using an averaged count of letter digram and trigram frequencies[283] to create nonwords does not always yield accurate approximations. Positional information of the letters within the word[284] needs to be taken into account.

Experience shows that over time developers learn to recognize the *style* of identifier spellings used by individuals, or development groups. Like the ability to recognize the *wordlikeness* of character sequences, this is another example of implicit learning. While studies have found that training in a given application domain affects people's performance with words associated with that domain, how learning to recognize an identifier naming *style* affects reader performance is not known. [] letter patterns implicit learning [782] words domain knowledge

6.2.4 Memory capacity limits

identifier STM required

This subsection discusses short-term memory limitations. Long-term memory limitations are an indirect cause of some of the usability issues discussed under other subsection headings in this usability section. The issue of some identifiers being available in LTM, or migrating to LTM during the process of comprehending source code, is not considered further here.

Short-term memory has a number of components, each having different capacity characteristics. The one of immediate interest here is the phonological loop, that is capable of holding approximately two seconds of sound. [] memory developer [] phonological loop

Identifiers appear in a number of different source code constructs, most commonly in expressions (the contents of comments are not considered here). When reading an expression, the identifiers it contains are components of a larger whole. One method of reducing the possibility of readers exceeding their short-term memory limit when attempting to comprehend expressions is to minimize the time taken to say (the internal spoken form in a persons head) each identifier. However, short-term memory capacity is not always the main consideration in selecting an identifier spelling. [782] optimal spelling identifier

Measuring the amount of time occupied by the verbal form of a word in short-term memory is a nontrivial task[228] (e.g., is it the time taken to say the word in isolation, say the same word repeatedly, or say the word in the context of a list of other words; where *say* is the form spoken internally in the mind, not the spoken form that vibrates the air and can be heard by others).

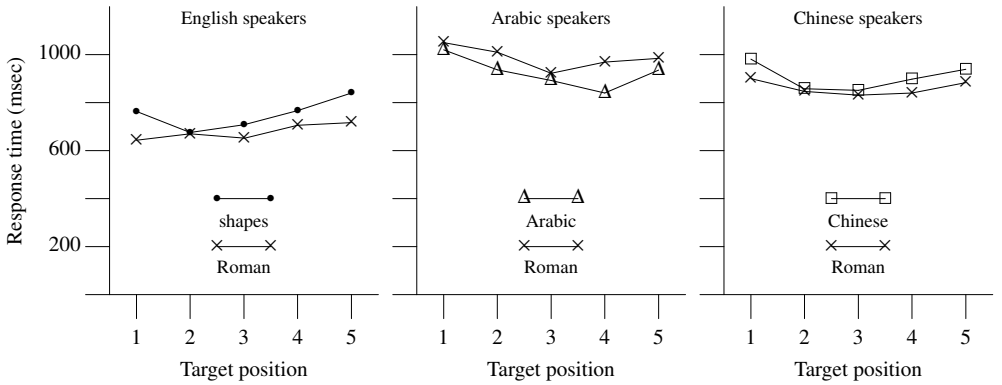


Figure 28: Mean response time (in milliseconds) for correct target detection as a function of the position of the match within the character sequence. Adapted from Green and Meara.[127]

Calculating the short-term memory requirements needed to hold information on words represented using a logographic writing system is more complicated than for alphabetic writing systems. A study by Hue and Erickson[151] found that literate Chinese subjects represented frequently occurring Chinese characters in verbal form, while low frequency characters were held in visual form.

logographic 782

6.3 Visual usability

The three main, identifier-related visual operations performed by developers—detailed reading, skimming, and searching —are discussed in the following subsections. The first subsection discusses some of the visual factors involved in extracting information from individual words. The general subject of human visual processing is discussed elsewhere.

vision early

6.3.1 Looking at a character sequence

In what order do people process an individual word? Do they start at the leftmost character and move their eyes successively to the right? Do they look at the middle of the word first and progressively work outwards? Do they look at the start and end of the word first, before looking at the rest of the word, or some other order of processing the character sequence?

orthography 782

- A study by Green and Meara[127] investigated the effects of orthography on visual search. Subjects (native English, Spanish, Arabic, or Chinese speakers) were asked to search for a character (Roman letter, Arabic character, or Chinese logograph) in a sequence of five characters (of the same kind). None of the results showed any trade-off between speed and accuracy (which varied between 5–8%). The results (Figure 28) for English and Spanish speakers were very similar—an upward sloping *M* response curve when searching letters and a *U* response curve when searching shapes (with the response slowing as the match position moves to the right). For Arabic speakers there was a *U* response curve for both Arabic and Roman characters (with the response slowing as the match position moved to the left; Arabic is read right-to-left). For Chinese speakers there was a *U* response curve for both Chinese and Roman characters (there was no left or right position dependency).

  An earlier study[128] comparing the performance of English children and adults found that performance improved with age and that left-to-right processing became more established with age.

identifiers Greek readers

- A study by Chitiri and Willows[55] compared how readers of English and Greek paid attention to different parts of a word. Greek primarily uses inflections to denote semantic and syntactic relationships. For instance, one or more characters at the end of a noun can indicate gender, number (singular, plural), and the case (nominative, genitive, accusative, vocative). These letters at the end of a word carry important information, and it is to be expected that experienced readers of Greek will have learned

to pay more attention to the ends of words that readers of languages that are not so inflective, such as English. The results showed that Greek readers tended to pay more attention to the ends of words than English readers.

- A study by James and Smith[162] asked subjects to search a string of letters for a designated letter. The results showed that for words, in certain cases, vowels were located more quickly than consonants (there was no difference in performance for nonwords). It was proposed that the difference in performance was caused by the position of vowels in words being more predictable than consonants in English. Subjects were using their knowledge of English spelling to improve their search performance. Searching words in all uppercase or all lowercase did not affect the results.[161]

- A review by Lukatela and Turvey[204] discussed research involving Serbo-Croatian readers. This language has two alphabets, one based on a Cyrillic alphabet and the other on a Roman one. Before the breakup of Yugoslavia, schoolchildren in the eastern half of the country learned the Cyrillic alphabet first, followed by the Roman. The order was reversed for schoolchildren in the western half of the country. Some letters were common to both alphabets but had different pronunciations in each. For instance, *potop* could be pronounced */potop/*, */rotop/*, */potor/*, or */rotor/* (two of which represented words—*deluge* in the Roman form, or *rotor* in the Cyrillic form).

- A study by Herdman, Chernecki, and Norris[141] measured subjects' response time and error rate when naming words presented in either lowercase or cAsE aLtErNaTeD form. The words also varied in being either high/low frequency or having regular/irregular spellings. Case alternation slowed response time by approximately 10%. However, it almost doubled the error rate for regularly spelled words (1.8% vs. 3.5% for high-frequency and 5.3% vs. 8.5% for low-frequency) compared to use of lowercase. The cAsE aLtErNaTiOn used in this study is probably more extreme than that usually seen in identifier spellings. As such, it might be considered an upper bound on the performance degradation to be expected when case alternation is used in identifier spelling. A study by Pring[259] investigated subjects' performance when presented with words using different case. Subjects saw letter sequences such as *CHurCH* and *ChuRCH*. The results showed an increase in error rate (4.4% vs. 2.7%) when a difference in case occurred across grapheme boundaries (a British English speaker might divide *CHURCH* into the graphemes *CH*, *UR*, and *CH*). No difference was found in the error rate for nonwords.

[782] grapheme

6.3.2 Detailed reading

Several studies[171] have found that people read prose written using lowercase letters more quickly (approximately 7%) than prose written using uppercase letters. There are a number of reasons for this, including: (1) when proportional spacing is used, a greater number of lowercase characters, compared to uppercase, fall within the visual field allowing larger chunks to be processed per saccade;[20] (2) words appearing in lowercase have a more distinctive shape to them, which is information that enables readers to make more accurate guesses about the identity of a word; and (3) readers have had more practice reading lowercase.

Studies of subjects' eye movements while reading have found that irregular or salient letter sequences at the beginning of a word[157, 321] cause the eye's initial landing position to be closer to the beginning of the word. However, the initial landing position was not affected by having the salient letters (those needed to disambiguate a word from other words) in the second half of a long word (Finnish words containing 10 to 13 characters).[158] Both of these findings are consistent with the Mr. Chips model of eye movement. Word predictability has been found to have little influence on the initial landing position.[261]

[] Mr. Chips

There are many differences between reading prose and reading source code. For instance, prose often has a narrative style that allows readers to progress through the material sequentially, while source code rarely has a narrative type and readers frequently have to refer back to previously read material. Whether these constant interruptions reduce the performance advantage of lowercase letters is not known. The extent to which any of these letter–case performance factors affect source code reading performance is not known.

```
#include <string.h>

#define MAXIMUM_CUSTOMER_NUMBER_LENGTH 13
#define VALID_CUSTOMER_NUMBER           0
#define INVALID_CUSTOMER_NUMBER         1

int check_customer_number_is_valid(char possibly_valid_customer_number[],
                                   int *customer_number_status)
{
int customer_number_index,
    customer_number_length;

*customer_number_status=VALID_CUSTOMER_NUMBER;
customer_number_length=strlen(possibly_valid_customer_number);
if (customer_number_length > MAXIMUM_CUSTOMER_NUMBER_LENGTH)
   {
   *customer_number_status=INVALID_CUSTOMER_NUMBER;
   }
else
   {
   for (customer_number_index=0; customer_number_index < customer_number_length; customer_number_index++)
      {
      if ((possibly_valid_customer_number[customer_number_index] < '0') ||
          (possibly_valid_customer_number[customer_number_index] > '9'))
         {
         *customer_number_status=INVALID_CUSTOMER_NUMBER;
         }
      }
   }
}
```

Figure 29: Example of identifier spellings containing lots of characters. Based on an example from Laitinen.[187]

visual skimming

6.3.3 Visual skimming

The number of characters in each identifier that appear in the visible source affects the visual appearance of any construct that contains it. As Figure 29 shows, the use of relatively long identifiers can affect the visual layout of constructs that reference them. The layout issues associated with this kind of usage are discussed elsewhere.

expression visual layout
declaration visual layout
statement visual layout
identifier visual search

6.3.4 Visual search

The commonly chosen, by readers, method of locating occurrences of an identifier in the code visible on a display is visual search. Given such behavior, visual search usability might be defined in terms of selecting an identifier spelling that minimize the probability of a search failing to locate an instance of the searched-for target identifier, or incorrectly classifying an instance as the target.

While there have been a considerable number of studies[241] investigating visual search, most have involved searching for objects of different shapes and colors rather than words.

Visually source code appears as an ordered sequence of lines. In many cases a complete declaration or statement appears on a single line. The reason for searching the source and the kind of information required can affect what is considered to be the best search strategy; for instance, skimming the characters in a similar order to the one used during detailed reading, scanning down the left edge of the source looking for an assigned-to object, or looking at conditional expressions to see where an object is tested. Your author does not know of any studies that have investigated this issue. The following discussion therefore has to been based primarily on studies using character sequences from other task domains:

- A study by Vartabedian[317] investigated search times for words presented on a CRT display. Subjects were asked to locate a word among a list of 27 words (either all lowercase or all uppercase) arranged as three columns of nine rows (not forming any meaningful phrases or sentences). The average search

time was less (approximately 13%) for the uppercase words.

- A study by Phillips[248] investigated the effect of letter case in searching for words on paper maps. The results showed that performance was best when the words consisted of an uppercase letter followed by lowercase letters (all uppercase resulted in the worst performance). A second study[249] investigated the eye fixations used in searching for words in maplike visual displays. The order in which subjects fixated words and the duration of the fixation was measured. The visual similarity between the word being searched for and the other words was varied (e.g., common initial letter, same word shape based on ascenders and descenders, and word length). The results showed that differences in these visual attributes did not affect eye movements—subjects tended to fixate a word, then a word close to it, and so on. (Some subjects worked from the top-down in a zigzag fashion, others worked clockwise or anti-clockwise around the display.) The duration of the fixation was affected by the similarity of the word being searched for. Objects of similar size and color to words were fixated less often than words, unless the resemblance was very close.
- A study by Flowers and Lohr[101] asked subjects to search for words consisting of either familiar English three-letter words or nonword trigrams with similar features in a display. The time taken, and error rate, for subjects to specify whether a word was or was not present in a display was measured. Distractor words with high similarity to the searched-for word were created by permuting the letters in that word (e.g., *BOY*, *BYO*, *OBY*, *OYB*, and *YBO*). Medium-similarity distractor words contained one letter that was the same as the searched-for word and low-similarity shared no letters. The results showed a significant difference in speed of search with high-similarity nonword distractors. In this case word searches were 30% to 50% faster than nonword searches. This performance difference dropped to 10% to 20% when the distractors were medium-similarity nonwords. There was little difference in performance when the distractors had low similarity or were words. The error rates were not found to be dependent on the distractors.
- A study by Karlin and Bower[167] investigated whether subjects could categorize a word semantically before they precisely identified the word. The time taken and error rate for subjects to specify whether a word was or was not present in a display was measured. The searched-for word either belonged to the same category as the other words displayed, or to a different category. For instance, the name of a color, say *PINK*, might be displayed with the names of other colors or the names of trees. The results showed that as the number of distractor words increased subjects increased their use of category information to improve search time performance (searching for a word in a different category was nearly a third faster when searching among six items than when it was in the same category). The error rates were not found to be category-dependent. Karlin and Bower proposed that comparison involved two kinds of tests, performed in parallel—a categorization test and a perceptual feature test. In those cases where a comparison requires a lot of perceptual resources because two items are perceptually similar, the categorization test may complete first. Flowers and Lohr explained their results in terms of categories, words, and nonwords. Searching for a word amongst nonword distractors that share the same letters imposes a high perceptual load because of overlap of features, and the categorization comparison completes first. When the distractors share a single letter, the difference is less pronounced.

When searching for an identifier, it is possible that mistakes will be made (either failing to locate an identifier that is present, or incorrectly matching the wrong identifier). The issue of identifier confusability is discussed elsewhere. [] identifier confusability

6.4 Acoustic usability

A number of factors contribute toward the acoustic usability of a character sequence. Memory and confusability acoustic factors are discussed in earlier subsections. Here, generating a pronunciation for a character sequence and meanings suggested by sounds (phonetic symbolism) are discussed.

Phonological codes (the basic sounds of a language) have been found to play an important role in access- [782] phoneme

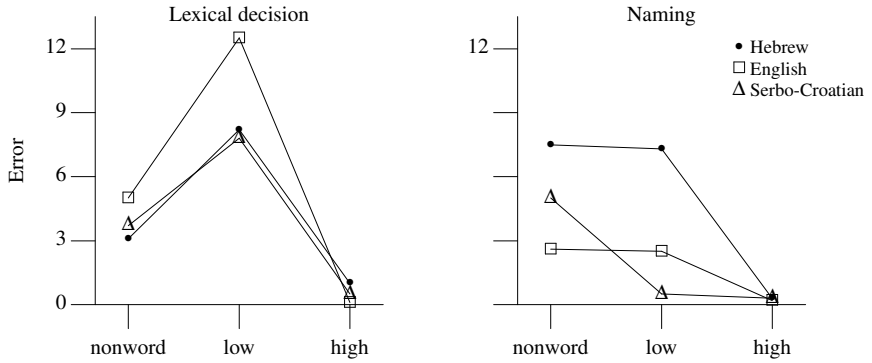


Figure 30: Error (as a percentage of responses) for naming and lexical decision tasks in Hebrew, English, and Serbo-Croatian using high/low frequency words and nonwords. Adapted from Frost, Katz, and Bentin.[107]

logographic 782

ing the semantic information associated with words written in nonlogographic scripts (some researchers believe they are obligatory,[102, 314] while others believe there is a single mechanism, e.g., connectionist models[254]). It was once thought that, because of their pictorial nature, logographs did not evoke phonological codes in readers. Studies[301] have found that phonographic codes also play a role in reading logographic scripts.

word pronounceability

6.4.1 Pronounceability

identifier wordlikeness

All character sequences can be pronounced in that the individual characters can be pronounced one at a time. The term *pronounceable* is commonly applied to character sequences that are wordlike and can be converted to a spoken form using the grapheme-to-phoneme conventions of a natural language familiar to the reader. In some languages (e.g., Japanese kana and Greek) virtually any combination of characters is pronounceable. This is because each character represents an individual sound that is not significantly affected by adjacent the characters.

'Twas brillig, and the slithy toves Did gyre and gimble in the wabe: All mimsy were the borogoves, And the mome raths outgrabe. Lewis Carroll

In this subsection pronounceability is measured in terms of the ease with which a reader is able to convert a sequence of characters to a spoken form (pronounceability could also be defined in terms of minimizing information content). Whether the spoken form used by the reader is the one intended by the author of the source code is not of concern here.

Abbreviating words removes what appear to be redundant characters. However, these characters are needed by readers if they are to recognize the graphemes their prior experience has trained them to expect (unless the reader recognizes the abbreviation and internally uses the word it represents). Abbreviations thus reduce pronounceability.

characters mapping to sound identifier information extraction abbreviating identifier

Experience shows that developers use a number of techniques to generate a spoken representation of character sequences. In some cases the sounds of the individual characters are used. In other cases developers mentally add characters (vowels and sometimes consonants) to what would otherwise be a nonpronounceable nonword (for abbreviations this is often the unabbreviated word); the sound used evolves over time, particularly when the character sequence is used in a spoken form between developers. Some developers simply take pleasure in inventing sounds for character sequences.

orthography 782

orthography 782

- A study by Frost, Katz, and Bentin[107] investigated the influence orthographic depth had on visual word recognition. Subjects were native speakers of Hebrew (a very deep orthography), Serbo-Croatian (a very shallow orthography), and English (an orthography somewhere between the two). The tasks involved word naming or making a word/nonword lexical decision and the response times and error rates were measured. The results were consistent with the idea that readers of shallow orthographies generate the pronunciation directly from the words, while for readers of deeper orthographies it is derived from an internal lexicon. The error rates are shown in Figure 30

Brain imaging studies[242] have found that Italian speakers (shallow orthography) activate different parts of their brain while reading compared to English speakers (deep orthography).

- A study by Lange and Content[193] analyzed the grapheme-to-phoneme correspondence of French words. (The results confirmed the view that this was quite predictable; phoneme to grapheme correspondence was not analyzed and is considered to be much less predictable.) They then used the results of this analysis to measure French speakers performance when reading words aloud. The selected words had either low/high grapheme frequency (number of occurrences in the corpse, independent of phoneme mapping) or low/high grapheme entropy (a measure of the number of different phonemes, and their frequency, a particular grapheme could be mapped to). They found that time to name a word and error rate did not vary significantly with grapheme frequency. However, there was a significant difference in error rate (but not time to name the word) when comparing words with a low/high grapheme entropy. This provided experimental evidence that more naming errors are made for words containing graphemes having many possible pronunciations than those having fewer possible pronunciations.

- A study by Rey, Jacobs, Schmidt-Weigand, and Ziegler[263] asked subjects to identify words (using matching native English and French words and subjects) containing five letters. These five-letter words contained either three, four, or five phonemes (e.g., for English: TEETH */tiT/*, BLEAT */blit/*, or BLAST */bl#st/* respectively). Subjects' response time and error rate were measured. The results showed that subjects' performance improved (slightly faster response time and decreased error rate) as the number of phonemes increased (except for high-frequency French words). A study by Rastle and Coltheart[260] found the same behavior for nonwords in both their DRC model of word naming and human subjects. The results from the DRC model suggested that reader expectations were causing the difference in performance. As each letter was successively processed, readers used it and previous letters to create a phoneme. In many cases these phonemes contained a small number of letters (perhaps just one) and readers started processing based on an expected common case. For instance, the letter *P* often represents the phoneme */p*; however, the following letter may create a multi-letter phoneme, and the previous processing on creating a pronunciation needs to be partially undone and processing started again (e.g., when the letter *P* is followed by *H*, the phoneme */f/* needs to be used). In the preceding studies it is not that the five phoneme letter sequences are processed more quickly, but that the three phoneme sequences are significantly slowed by additional cycles of processing, undoing, and reprocessing.

[] Word recognition models of

- A study by Stanovich and Bauer[290] showed that the regularity of spelling-to-sound correspondence affected performance; for instance, the regular pronunciation of *_INT* occurs in *MINT*, *HINT*, *DINT*, and an irregular pronunciation in *PINT*. Their study showed that regularity had some impact on the time taken to name a word (552ms vs. 570ms) and on a lexical decision task (623ms vs. 645ms).

- A study by Stone, Vanhoy, and Orden[295] showed that not only did spelling-to-sound regularity (feedforward consistent) affect performance, but sound-to-spelling regularity (feedback consistency) was also a factor. Visual word perception is a two-way street. For instance, the sound */_ip/* can have either of the spellings that occur in *HEAP* and *DEEP*. The results showed, in a lexical decision task, an error rate of 3.9% for feedback-consistent and 9.8% for feedback-inconsistent words. There was no difference in error rate for nonwords.

6.4.1.1 Second language users

The following discussion and studies are based on having English as the second language. The results point to the conclusion that encoding strategies used by a person in their first language are transferred to English. This behavior can result in a different phonological encoding being formed, compared to the set of encodings likely to be used by native English speakers, when an unknown character sequence is encountered. The issue of pronouncing characters unfamiliar to the reader is discussed elsewhere.

[] reading characters unknown to reader

- A study by Holm and Dodd[147] investigated how phonological awareness skills acquired while learning to read and write a first language were transferred to learning to read and write a second language. The subjects were students at an Australian university who had first learned to read and write their native language in China, Hong Kong, Vietnam, and Australia. The characters learned by the Chinese and Hong Kong subjects were the same—logographs representing a one-syllable morpheme. The Vietnamese and Australian subjects had been taught using an alphabetic writing system, where a mapping to phonemes existed. As an aid to the teaching of reading and writing, China introduced an alphabetic system using Latin symbols called *pinyin* in 1959. No such system had been used by the subjects from Hong Kong. One of the tasks involved creating spoonerisms from pairs of written words (e.g., *dark ship* ⇒ *shark dip*). This task requires segmenting words based on phonological rules not letter rules. The results of the study showed that the performance of Hong Kong subjects was significantly worse than the other subjects in these kinds of phonological-based tasks. While they were proficient readers and writers of English, they had no ability to associate a written word with its pronunciation sound unless they had previously been told of the association. Holm and Dodd compared the performance of the Hong Kong students with phonological dyslexics that have been documented in the research literature. Interviews with the Chinese subjects found that one of the strategies they used was to "thinking how to write the word in pinyin". Some of the Hong Kong students reported being able to recognize and use written words by looking up their definitions in a dictionary. However, on hearing one of these words in a lecture, they were not able to make the association to the written form they had previously encountered unless they saw it written down (e.g., in the notes written by the student sitting next to them in class). The results of this study showed that for some developers identifier pronounceability is not an issue because they don't pronounce them.

Spoonerisms

- A study by Koda[175] investigated the impact of subjects' first language (Arabic, English, Japanese, or Spanish; who had had at least six years of English instruction) on their phonological encoding strategies for various kinds of character sequences. Subjects were shown five character sequences followed by a probe (one of the five character sequences). They had to specify which character sequence followed the probe. The character sequences were either phonologically similar English nonwords, phonologically dissimilar but visibly similar English nonwords, unpronounceable sequence of letters, pronounceable Japanese Kanji (logographs that were either visually similar or dissimilar), unpronounceable Japanese Kanji, or Sanskrit (a logography unfamiliar to all subjects). The results showed that subjects, independent of language background, performed better when phonologically dissimilar English pronounceable nonwords were used (phonological similarity causes interference in working memory) and the performance of Japanese subjects was not significantly affected by the use of unpronounceable English nonwords. The results from the Japanese lists showed that performance among the three non-Japanese subjects did not differ significantly. There was no significant difference between any group of subjects on the Sanskrit lists.

acoustic confusability

- A study by Furugori[112] found that the typical spelling mistakes of Japanese users of English reflected the lack of differentiation, made in Japanese, of some English phonemes (e.g., Japanese has no */θ/* phoneme as in *th*ink, which is heard as */s/*, leading to *thunderstorm* being spelled *sanderstorm*).

identifier

6.4.2 Phonetic symbolism

phonetic symbolism

Studies have found that native speakers of English have the ability to guess the meanings of words from unfamiliar languages with greater than chance probability. For instance, a study by Brown, Black, and Horowitz[42] asked subjects (English speakers unfamiliar with the other languages used in the study) to match English antonyms against the equivalent Chinese, Czech, and Hindi pairs of words (58.9%, 53.7%, and 59.6% pairs were correctly matched, respectively). It has been proposed that there exists a universal phonetic symbolism. This proposal implies that sounds tend to have intrinsic symbolic connotations that are shared by humans and that traces of these sound-to-meaning linkages survive in all natural languages.

antonym 782

feeling of knowing

A study by Koriat[176] found that the stronger a subject's feeling of knowing for a particular antonym

pair, the higher the probability of their match being correct (English antonyms were required to be matched against their Thai, Kannada, and Yoruba equivalents).

The symbolism commonly associated with certain word sounds has been known about for many years; for instance, that words with *K* or *P* sounds are funny (chicken, pickle, cucumber, and porcupine). Advertisers also make use of the sound of a word when creating names for new products. A study by Schloss[270] showed that 27% of the top 200 brands of 1979 began with *C*, *P*, or *K*; 65% began with *A*, *B*, *C*, *K*, *M*, *P*, *S*, or *T* (with less than a 5% probability of this occurring by chance).

A study by Magnus[207] looked at monosyllabic words in English. The results showed that words containing a given consonant fell within much narrower semantic domains than would be expected if there were no correlation between phonology and semantics. The term *phonesthemes* was defined to refer to a sound sequence and a meaning with which it is frequently associated. An example of a phonestheme is the English */gl/* in initial position being associated with indirect light.

Table 19: Words that make up 19 of the 46 words beginning with the English */gl/* of the monomorphemic vocabulary (Note: The others are: globe, glower, glean, glib, glimmer, glimpse, gloss, glyph, glib, glide, glitter, gloss, glide, glissade, glob, globe, glut, glean, glimmer, glue, gluten, glutton, glance, gland, glove, glad, glee, gloat, glory, glow, gloom, glower, glum, glade, and glen). Adapted from Magnus.[207]

Concept Denoted	Example Words
Reflected or indirect light	glare, gleam, glim, glimmer, glint, glisten, glister, glitter, gloaming, glow
Indirect use of the eyes	glance, glaze(d), glimpse, glint
Reflecting surfaces	glacé, glacier, glair, glare, glass, glaze, gloss

Magnus also asked subjects (predominantly English speakers responding to a Web survey) to provide definitions for made-up words. For many of the words the particular definitions provided where limited to a small number of semantic domains, often with two domains accounting for more than half of the definitions.

6.5 Semantic usability (communicability)

Identifier semantic usability

A commonly specified coding guideline is that *meaningful* identifiers be used. This subsection discusses the topic of shared semantic associations (the sharing is treated as occurring between the original creator of an identifier spelling and subsequent readers of it), a more technical way of saying *meaningful*.

[] semantic networks

Developers have an existing semantic net in their head, which maps between character sequences and various kinds of information. This net has been continually building up since they first started to learn to read and write (as such, very little of it is likely to be directly related to any computer-related applications or software development). This existing semantic net is of major significance for several reasons, including:

- Making use of existing knowledge enables a small amount of information (an identifier's spelling) to represent a much larger amount of information. However, this is only possible when the original author and the reader share the same associations (or at least the ones they both consider applicable to the circumstances) for the same character sequences.
- Creating new memories is a time-consuming and error-prone process; making use of existing ones can be more efficient.
- A large amount of the information that it contains is not explicitly available. Developers often apply it without consciously being aware of the extent to which their decisions are driven by culturally (in its broadest sense) specific learning.

[] implicit learning

Given the current state of knowledge about the kinds of semantic associations made by groups of people working in a common field, even predicting how developers educated in the same culture might communicate effectively with each other is very difficult, if not impossible. Predicting how developers educated in different cultures might communicate, via identifier spellings, creates even more layers of uncertainty.

However impossible prediction might be at the time of writing, it is a problem that needs to be addressed. The following subsections discuss some of the issues.

6.5.1 Non-spelling related semantic associations

The need to recall information about an identifier is often prompted by its being encountered while reading the source. (A developer may ask another developer for information on an identifier and provide a spoken rendition of its spelling, but this usage is much less common.) The context in which the identifier occurs can often be used to deduce additional information about it. For instance, the identifier is a typedef name (it occurs in a list of declaration specifiers), it has arithmetic type (it occurs as the operand of a binary multiplication operator), or designates an array or function (it occurs immediately to the left of a particular kind of bracket).

Although often providing important information to readers of the source, these nonspelling-related semantic associations are not discussed further here.

6.5.2 Word semantics

What does it mean to know a word? Richards[265] gave the following answer, which has been very influential in the field of vocabulary acquisition:

1. The native speaker of a language continues to expand his vocabulary in adulthood, whereas there is comparatively little development of syntax in adult life.
2. Knowing a word means knowing the degree of probability of encountering that word in speech or print. For many words, we also *know* the sort of words most likely to be found associated with the word.
3. Knowing a word implies knowing the limitations imposed on the use of the word according to variations of function and situation.
4. Knowing a word means knowing the syntactic behavior associated with that word.
5. Knowing a word entails knowledge of the underlying form of a word and the derivatives that can be made from it.
6. Knowing a word entails knowledge of the network of associations between that word and the other words in language.
7. Knowing a word means knowing the semantic value of a word.
8. Knowing a word means knowing many of the different meanings associated with the word.

The following discussion involves what was once thought to be a semantic effect on word naming performance—*imagineability*. Subsequent studies have shown that age of acquisition is the primary source of performance difference. This work is discussed here because many sources cite imagineability effects as a word-handling performance issue. A study by Strain, Patterson, and Seidenberg[296] asked subjects to read aloud words that varied across high/low frequency, regular/irregular spelling–sound correspondence, and high/low imagineability (as judged by 40 members of staff at the author's workplace; examples of high imageability included *corkscrew* and *sparkling*, while words with low imageability included *naive* and *presumption*). The results showed that on one case (low-frequency, irregular spelling–sound correspondence low imageability) the error rates were significantly higher (14–19% vs. 0–3%) than the other cases. It was proposed that the semantic clues provided by imageability provide a performance benefit that is only significant when processing irregularly spelled low frequency words. However, a later study by Monaghan and Ellis[221] also took a words age of acquisition into account. The results showed that age of acquisition had a significant effect on word naming performance. Once this effect was taken into account the apparent effects of imageability disappeared. It was pointed out that words learned early in life tend to be less abstract than those learned later. It is age of acquisition that is the primary effect.

age of acquisition 782

6.5.3 Enumerating semantic associations

semantic associations enumerating

How can the semantic associations evoked by a word be enumerated? One method is to enumerate the list of words that are considered to be related, or similar, to it. To this end this subsection discusses some of the algorithms that have been proposed for measuring the semantic relatedness, or similarity, of two words. It is based on the review by Budanitsky.[44] To quote from its opening sentence "Is *first* related to *final*? Is *hair* related to *comb*? Is *doctor* related to *hospital* and, if so, is the connection between them stronger than that between *doctor* and *nurse*?"

The methods of measuring similarity proposed by researchers can be broadly divided into two groups. The context-free methods do not consider the context in which the two words occur. Some compendium of words (e.g., a dictionary or thesaurus) is used to provide the base information. The context-sensitive methods consider the context in which the two words are used.

6.5.3.1 Human judgment

word similarity human judgment

One way to obtain a list of words associated with a particular word is human judgment. Studies of human semantic memory, by cognitive psychologists, often make use of word association norms. However, most of these studies use a small set of words. For instance, a study by Burke, Peters, and Harrold[47] measured associations for 113 words using 80 young (mean age 21.7) and 80 older (mean age 71.6) subjects. There have been two studies, using English speakers, that have collected associations for a large number of words.

A long-term study (it started in 1973) by Nelson, McEvoy, and Schreiber[235] created the University of South Florida word association, rhyme, and word fragment norm database by collating the nearly three-quarters of a million responses to 5,019 stimulus words from more than 6,000 subjects. Subjects (students at the university) were given lists of words (approximately 100 in a booklet, with 25–30 per page) and asked to write the first word that came to mind that was meaningfully related or strongly associated to each of these words. For example, given *book*, they might write *read*.

Nelson et al. compared their results with those obtained by Kiss, Armstrong, Milroy, and Piper[173] in the UK. They found substantial differences between the results and suggested that these were caused by cultural differences between Florida and the UK. For example, the most frequent responses of the Florida subjects to the word *apple* were *red* and *orange* (the fruit), with *tree* and *pie* being given relatively infrequently.

A study by Steyvers[292] used the Nelson et al. word association data to build a *Word Association Space* (which was claimed to have psychological relevance). The results showed that this space was a good predictor of similarity rating in recognition memory, percentage correct responses in cued recall, and intrusion rates in free recall.

6.5.3.2 Context free methods

The advantage of using existing sources of word associations, such as dictionaries, is the large amount of information they provide about general word associations. The disadvantages with using these sources of information is that the word associations they contain may not include specific, technical uses of words or even have some of the words being analyzed. The availability of large volumes of written text on the Internet has shown that the primary definitions given to words in dictionaries are often not the definitions commonly used in this material (although publishers are starting to produce dictionaries[62] based on measurements of common word usage). Also a human-written dictionary in computer readable form may not be available (the calculations for the original thesaurus study[226] were done by hand because online material was not available to the authors).

The following are some of the studies that have used context-free methods:

- A study by Kozima and Furugori[177] used a subset of the *Longman Dictionary of Contemporary English*[200] (*LDOCE*) to build a semantic network. This dictionary uses a small (2,851 words) vocabulary to express the meanings of all the other words (more than 56,000) it defines.[15] This semantic

[15] This feature, along with the publisher making it available to researchers for a small fee, has made LDOCE widely used by language researchers.

network was used to calculate the similarity between words using spreading activation. Later work by Kozima and Ito[178] made context-sensitive measurements by performing what they called *adaptive scaling of the semantic space*. They gave the example of *{car, bus}* having the context *vehicle* and being closely associated with *taxi*, *railway*, *airplane*, and so on, while *{car, engine}* had the context *components of a car* and were closely associated with *tire*, *seat*, *headlight*, and so on. A change of context usually resulted in a change of distance between the same word pair.

- Ellman[96] created a tool to detect lexical chains (constructs used in calculating text similarity) in large texts, with the intent of extracting a representation of its meaning. Various kinds of semantic relations between the words classified in *Roget's Thesaurus* (a thesaurus is intended as an aid in finding words that best express an idea or meaning, while a dictionary explains the meaning of words) were used to compute lexical chains between pairs of words appearing in the texts.

semantic networks

6.5.3.3 Semantic networks

A semantic network consists of a set of nodes with connections, and arcs, between them. The nodes representing concepts and the arcs denoting relationships between the nodes (concepts) they connect.

WordNet

WordNet[16][99] is a semantic network whose design was inspired by current psycholinguistic theories of human lexical memory. It organizes English nouns, verbs, adjectives, and adverbs into synonym sets (synsets), each expressing one underlying lexical concept (Table 20). Edmonds[93] provides a detailed discussion (and a computational model) of the fine-grained meanings of near synonyms and the differences between them. For instance, all of the WordNet noun synsets are organized into hierarchies. At the top of the hierarchies are the following nine abstract concepts called *unique beginners*:

Table 20: WordNet 2.0 database statistics.

Part of Speech	Unique Strings	Synsets	Total Word-sense Pairs
Noun	114,648	79,689	141,690
Verb	11,306	13,508	24,632
Adjective	21,436	18,563	31,015
Adverb	4,669	3,664	5,808
Total	152,059	115,424	203,145

- Entity—that which is perceived or known or inferred to have its own physical existence (living or nonliving)
- Psychological feature—a feature of the mental life of a living organism
- Abstraction—a general concept formed by extracting common features from specific examples
- State—the way something is with respect to its main attributes: *the current state of knowledge*, *his state of health*, *in a weak financial state*
- Event—something that happens at a given place and time
- Act, human action, human activity—something that people do or cause to happen
- Group, grouping—any number of entities (members) considered as a unit
- Possession—anything owned or possessed
- Phenomenon—any state or process known through the senses rather than by intuition or reasoning

The arcs between nouns in synsets are defined by the following relations:

antonym

- Antonym—the *complement* relation; words of opposite meaning (e.g., hot-cold)
- Holonymy—the *has a* relation, the opposite of meronymy

holonymy

- Hypernymy—the *is a* relation (e.g., plant is a hypernym of tree) hypernymy
- Hyponymy—the *subsumes* relation, the inverse of hypernymy
- Meronymy—relationship between objects where one is a part of the other (e.g., sleeve is a meronym of coat, dress, or blouse) hyponymy
- Synonym—word with the same, or nearly the same meaning meronymy synonym

Steyvers and Tenenbaum[294] investigated the graph theoretic properties of the semantic networks created by WordNet, *Roget's Thesaurus*, and the associative word lists built by Nelson et al. The results showed that they had a small world[5] structure.

6.5.3.4 Context sensitive methods

The attributes associated with an unknown word can often be inferred from the context in which it occurs; for instance, the paragraph: "A bottle of tezgüino is on the table. Everybody likes tezgüino. Tezgüino makes you drunk. We make tezgüino out of corn." suggests that *tezgüino* is an alcoholic drink made from corn mash.

Context-sensitive methods obtain their information directly from a corpus of written material. These methods all assume the words occur in text written in a natural language. As such, they are not directly applicable to identifiers in source code. However, they do provide a possible mechanism for automatically obtaining word similarity information for specialist domains (e.g., by processing books and papers dealing with those domains).

The advantages of context-sensitive methods are that they are not limited to the words appearing in some predefined source of information and because the needed information is automatically extracted from a corpus they are sensitive to the associations made. The disadvantage of this method is that the corpus may not contain sufficient occurrences of particular words for an accurate evaluation of their associations to be calculated.

One of the most widely discussed context-sensitive methods is *Latent Semantic Analysis*, LSA.[191] The underlying idea is that the sum of all the contexts in which a given word does and does not appear provides a set of mutual constraints that determines the similarity of meanings of words and sets of words to each other. The process of extracting relations between words starts with a matrix, where each row stands for a unique word and each column stands for a context (which could be a sentence, paragraph, etc.). Each matrix cell holds a count of the number of times the word it represents occurs in the context it represents. Various mathematical operations are performed (to the uninitiated these seem completely disconnected from the problem at hand and for this reason are not described here) on the matrix to yield results (each word is mapped to a vector in an n-dimensional space, where n is usually around 300, and the similarity between two words is calculated from cosine of the angle between their respective vectors) that have been found to effectively model human conceptual knowledge in a growing number of domains. latent semantic analysis

LSA takes no account of word order ("dog bites man" and "man bites dog" are treated the same way), syntactic relations (no syntactic parsing of the text is performed), or morphology (*fast* and *faster* are treated as different words). This eliminates many of the practical difficulties experienced by other methods. This simplicity, along with the quality of its results has made LSA a popular choice for information-retrieval problems. [782] morphology

An obvious way to try to improve the quality of word similarity measures is to take their syntactic relationships into account. Lin[197] proposed a similarity measure that takes the grammatical relationship between the two words into account. The raw data from which this word information is extracted is a list of sentences. These sentences are broken down into dependency triples, consisting of two words and the grammatical relationship between them, within the sentence. For instance, the triples for the sentence "I have a brown dog" are: (*have* subj *I*), (*I* subj-of *have*), (*dog* obj-of *have*), (*dog* adj-mod *brown*), (*brown*

[16]Probably the most well-known semantic network in linguistics, and it is available for download from the internet.

adj-mod-of *dog*), (*dog* det *a*), (*a* det-of *dog*), given two words w and $w^{'}$, and the relation r. The similarity between two words is based on the amount of information contained in the commonality between them, divided by the amount of information in the description of them, and the amount of information, I, contained in the dependency tuple $\parallel w, r, w^{'} \parallel$ is given by:

$$I(w, r, w^{'}) = \log(\frac{\parallel w, r, w^{'} \parallel \times \parallel *, r, * \parallel}{\parallel w, r, * \parallel \times \parallel *, r, w^{'} \parallel}) \quad (3)$$

where $*$ matches all words.

Lin obtain his sentences from various online newspapers (a total of 64 million words). An English language parser extracted 56.5 million dependency triples, 8.7 million being unique. There were 5,469 nouns, 2,173 verbs, and 2,632 adjectives/adverbs occurring more than 100 times.

6.5.4 Interperson communication

Identifier spellings provide a delayed, one-way form of human communication. The original author decides on a spelling to use, often with the intent of it denoting meaningful information, and sometime later a second person reads it (code reviews offer an opportunity for other developers to perform the role of future readers, but they are not usually held for this purpose). This form of communication is significantly different from the collaborative process that underlies most human communication. For instance, a study by Clark and Wilkes-Gibbs[58] showed how two people work together in the creation of agreed-on references (to complex shapes). The results also found that the number of words used decreased over successive trials (rearranging square and triangular paper cards to form complex shapes) as subject pairs learned from each other.

code reviews

The writers and readers of source code rarely get to take part in a collaborative communications process with each other. Furthermore, the original author may not even have other human readers in mind, when deciding on an identifier spelling. Authors may see themselves as communicating with the computer or communicating with themselves at some future date.

The following two subsections discuss the evolution of terminology groups by use in communicating among themselves and some of the issues involved in two people reaching the same conclusions about the semantic associations of a word or phrase.

6.5.4.1 Evolution of terminology

All but the smallest software development project will have more than one person working on it, although each particular piece of source code often has a single person working on it. While in many cases the number of people who actually write significant amounts of source is usually only a fraction of the total number of people on the project, there is invariably a set of linguistic conventions (including a terminology) that evolves and is shared by a large percentage of the members of a project.

The characteristics of the evolution of linguistic conventions that occur in groups are of interest to the extent that they affect the performance of subsequent readers of the source.

A study by Garrod and Doherty[116] investigated the establishment of linguistic conventions in two different kinds of groups. Two people had to work together to solve a maze game in which the information needed was distributed between them (the only way of communicating the needed information was by questioning and answering each other; responses were recorded by the experimenter). In one group (five pairs) the same two people always worked together, while in the other group everybody (10 people) eventually got to be paired with everybody else in the group. Each subject played nine 10-minute games with the maze randomly generated for each game.

The results showed that isolated pairs of subjects had much higher levels of inter-speaker coordination (measured by categorizing each information exchange that took place between subjects while solving a particular maze, and counting the exchanges in each category) compared to the community pairs during the first few games. However, by the time six games had been played, this situation had reversed, with the community pairs having significantly more inter-speaker coordination.

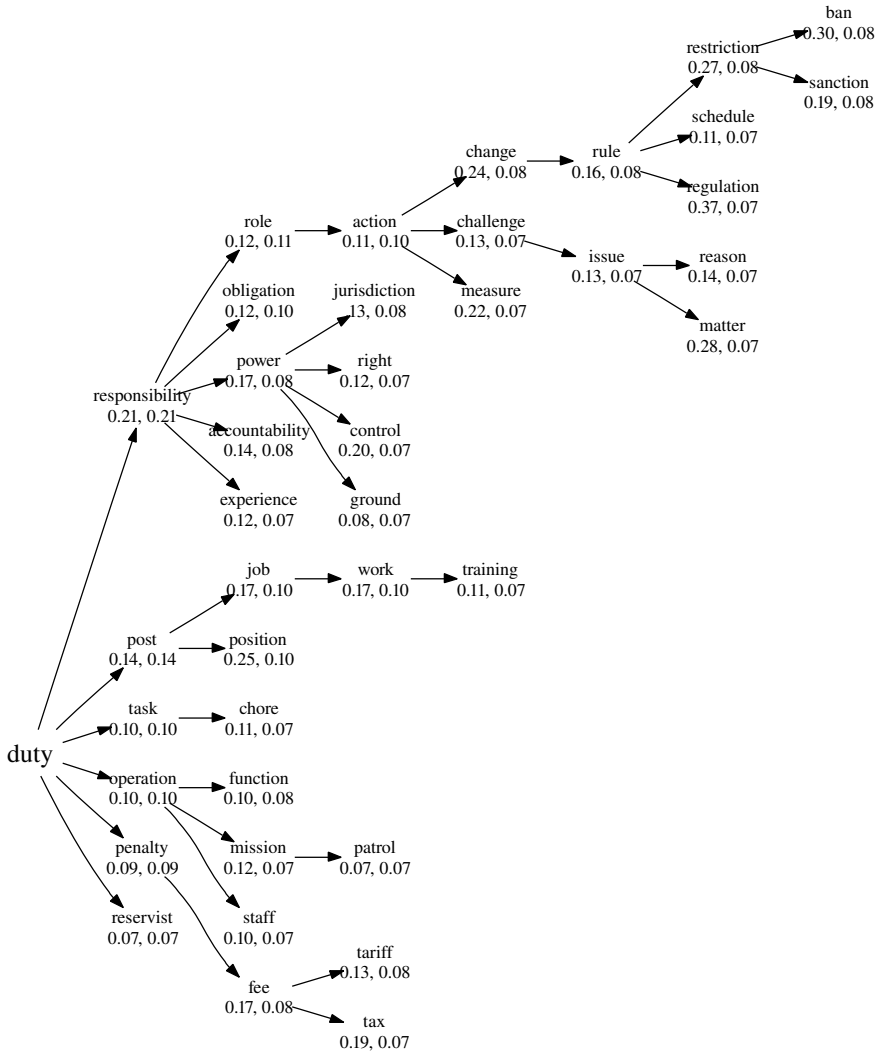


Figure 31: Similarity tree for *duty*. The first value is the computed similarity of the word to its parent (in the tree), the second value its similarity to *duty*. Adapted from Lin.[197]

The way in which the two groups coordinated their descriptions differed. The isolated pairs used descriptions that were specific to a particular position in the solution and specific to themselves. Once the community pairs became established through overlapping interactions, they began to coordinate as a group. The constraints on their descriptions became global ones that apply to communities in general (i.e., they were not able to make use of special cases previously agreed to between two individuals).

If the behavior seen in the Garrod and Doherty study also occurs in development groups, it is to be expected that groups of one or two developers are likely to evolve terminology that is much more tightly bound to their mutually agreed-on way of thinking than larger groups of developers.

6.5.4.2 Making the same semantic associations

The semantic usability of an identifier's spelling might be judged by the degree to which the semantic associations it creates reduces the effort needed to comprehend source code containing it. A randomly selected identifier spelling is extremely unlikely to create semantic associations having this property. Selecting an identifier's spelling to create the necessary associations is the solution. However, the decision on the spelling to use is judged by the semantic associations created in the original author's mind at the time it is selected. It cannot be assumed that the semantic associations of the original author are the ones most relevant to subsequent readers. This subsection discusses some of the issues involved.

Both the authors and readers of an identifier's spelling will make assumptions about how it is to be interpreted.

- *Authors* of source code will make assumptions about the thought process of subsequent readers. Explicit assumptions made might include: degree of familiarity with application domain (e.g., to simplify their task the original authors may have specified to management that competent people need to be hired and then proceed to work on the basis that this requirement will be met), or that they have read the documentation. Implicit assumptions might include: culture (no thought given to alternative cultures), degree of familiarity with the application domain (e.g., is familiar with certain terms such as *overload*, *spill* and abbreviations such as *fft*, *cse*, *sql*), or education of the reader (e.g., familiarity with mathematics to a certain level).
- *Readers* of the source code will make assumptions about the intent of the original author. An explicit assumption might be *trusting* the original author (e.g., "the software has been working well"). The reader may have limited time available and trust may seem like the best way of reducing the time they need to invest; whether this trust is justified is not the issue here. The implicit assumptions might include: the reader and author share common ground, that the original author had intentions about the identifiers used in a particular section of source (original authors are often seen as mythological figures; it is possible that no particular interpretation was intended), and degree of familiarity with the application domain.

Given the many different kinds of assumptions that different developers may make, people might wonder how readers can be expected to obtain helpful semantic information from identifier spellings. The answer is that often they aren't. There are two routes by which different people might arrive at similar semantic interpretations for an identifiers spelling:

1. *Shared knowledge.* Shared knowledge may occur through culture, natural language usage, or specific learning experiences. An example of where differences in shared knowledge produce differences in performance is the treatment of time in English and Mandarin Chinese. English predominantly treats time as if it were horizontal (e.g., "ahead of time", "push deadline the back"), while Mandarin often, but not always, treats it as being vertical (an English example of vertical treatment is "passes down the generations").[272] A study by Boroditsky[29] found that English subjects responded more quickly (by approximately 10%) to a question about dates (e.g., "Does March come before April?") if the

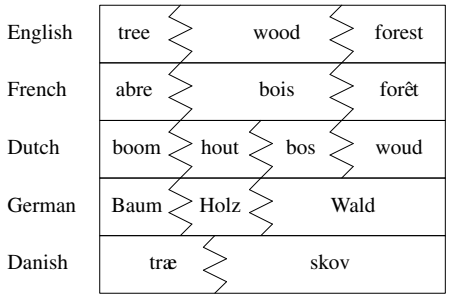


Figure 32: The relationship between words for tracts of trees in various languages. The interpretation given to words (boundary indicated by the zigzags) in one language may overlap that given in other languages. Adapted from DiMarco, Hirst, and Stede.[88]

previous question they had answered had involved a horizontal scenario (e.g., "X is ahead of Y") than if the previous question had involved a vertical scenario (e.g., "X is below Y"). These results were reversed when the subjects were Mandarin speakers and the questions were in Mandarin.

2. *Shared behavior.* Possible shared behaviors include effort minimization, universal grammar, and universal category formation. An example of how shared behavior can affect people's communication was shown by the results of a study by Beun and Cremers.[23] They gave pairs of subjects, a builder and an instructor, the task of building a replica of a building that was only visible to the instructor. The two subjects were seated at a table and could speak to each other and could see each others' hands, but there was no other mode of communication. A pile of blocks of different colors, shapes, and sizes, only visible to the builder, were provided. The block building had to be done on a plate visible to both subjects. Both subjects spoken conversation and hand gestures were recorded. The behavior predicted by Beun and Cremers is based on the *principle of minimal cooperative effort*, where the speaker and addressee not only try to say as little as possible together, they also try to do as little as possible. For instance, what features of an object should be used in a description? This principle suggests that people will prefer absolute (e.g., *black* or *square*) rather than relative (e.g., *darkest*, *longest*) features. This is because absolute features only require one object to be taken into account, while relative features requires comparison against other objects. The results found that 63% of referential acts used absolute features only, 19% used a combination of absolute and relative features, and 1% used relative features only. A pointing action, with the hands, was used in 18% of cases. The study also found evidence for several other hypotheses derived from this principle, including: (1) if the target object is inherently salient within the domain of conversation, reduced information is used; and (2) if the target object is located in the current focus area, only information that distinguishes the object from other objects in the focus area is used.

6.6 Abbreviating

abbreviating identifier

Natural language words are often used as the basis for identifier spellings. Having selected one or more words, developers sometimes decide to reduce the total number of letters in the final spelling. Some form of shortening occurs. This section discusses the issues associated with shortening words (the issue of source filename abbreviations is discussed elsewhere), these include:

[] file name abbreviations

Developers sometimes form an identifier by abbreviating a descriptive sentence (e.g., `first_elem_ptr` might be derived from *a pointer to the first element of an array*) However, complete sentences are rarely abbreviated in everyday life, while multiword phrases may eventually be replaced by an acronym (e.g., *Light Amplification by Stimulated Emission of Radiation* became *LASER* and eventually *laser*). For this reason there does not appear to be any published research on the creation of abbreviations from multiple words.

The following are some of the issues associated with using shortened forms:

word pronounceability

- Shortening is likely to change the amount of cognitive resources needed by readers of the source to process an identifier containing them. This is because abbreviation changes the distribution of character sequences, with infrequent or never-seen character pairs occurring more often. It may also remove any obvious grapheme-to-phoneme mapping, making it harder to create a pronunciation
- Fewer characters means less effort is needed to type the character sequence denoting an identifier
- Reducing the number of characters in an identifier can simplify the visual organization of source code (i.e., by removing the need needing to split an expression or statement over more than one line)
- While some abbreviations may have semantic associations for the original developer, these are often not understood or are forgotten by subsequent readers of the source. Such identifier spellings are then treated as a random sequence of characters

Word shortening can be studied by asking people to create shortened forms of words and phrases;[143, 297] by analyzing the shortened forms occurring in prose,[288] source code,[188] speech,[51] or from a purely information content point of view.[34] These studies investigated the form of the abbreviations created, not the circumstance under which people decide to create or use an abbreviation. A number of commonly occurring patterns to the shortened forms have been found, including:

- Vowel deletion—sometimes known as *contraction* (e.g., *search*⇒`srch` and *pointer*⇒`pntr`)
- Truncation of trailing characters (e.g., *audit* ⇒ `aud` and *catalog* ⇒ `cat`)
- A combination of vowel removal and truncation (e.g., *pointer*⇒`ptr` and *temporary*⇒`tmp`)
- Using the first letter of each word (e.g., *customer query number*⇒`cqn` and *first in first out* ⇒ `fifo`)
- Phonetic abbreviations—simply changing one grapheme to another that represents the same phoneme (e.g., *ph*⇒*f*) or digits may be used (e.g., *straight*⇒*str8*)

These abbreviations may in turn be concatenated together to create specialized instances (e.g., `tmp_cqn`, `cat_pntr`, and `srch4cqn`).

An abbreviation is generally a meaningless sequence of characters unless readers can decode it to obtain the original word. A study by Ehrenreich and Porcu[94] found that readers' performance in reconstructing the original word was significantly better when they knew the rules used to create the abbreviation (81–92% correct), compared to when the abbreviation's rules were not known (at best 62% after six exposures to the letter sequences). In practice imposing a fixed set of word abbreviation rules on the world's software developers is not realistic. Imposing a fixed set of abbreviation rules on the developers within a single development group is a realistic possibility, given automated checking to ensure conformance. However, calculating the likely cost/benefit of imposing such a set of rules is very difficult and these coding guidelines do not discuss the issue further.

The widespread use of phonetic abbreviations is relatively new. The growth in online chat forums and the use of text messaging via mobile phones has significantly increased the number of people who use and understand their use. The extent to which this user population intersects the set of creators of identifier spellings also continues to grow. Some people find this use of abbreviations irritating and irresponsible. These coding guidelines take the position that all character sequences should be judged on their ability to communicate information to the reader. Your author has not been able to find any published studies of the use of phonetic abbreviations and they are not discussed further here.

A study by Frost[106] investigated subjects' performance in reading words with missing vowels. The subjects were experienced readers of Hebrew, a language in which words are usually written with the vowels omitted. The time taken to name words was found to vary linearly with the number of vowels omitted. Missing vowels had no effect on lexical decision performance. One point to note is that there was only one word corresponding to each letter sequence used in the study. It is not known how reader performance would vary if there was more than one word matching a vowel-free written form.

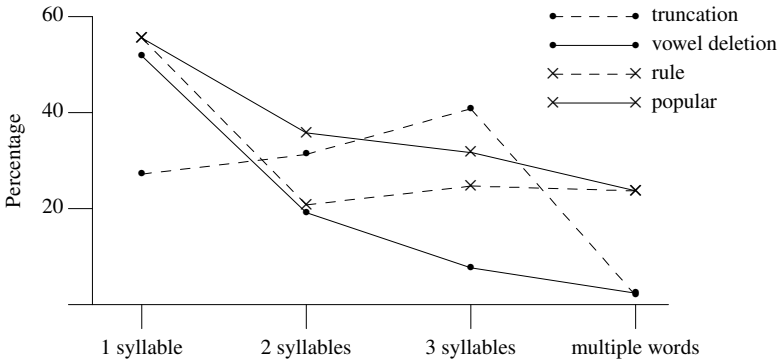


Figure 33: Percentage of abbreviations generated using each algorithm. The *rule* case was a set of syllable-based rules created by Streeter et al.; the *popular* case was the percentage occurrence of the most popular abbreviation. Based on Streeter, Ackroff, and Taylor.[297]

The results of two studies asking people to abbreviate the words and phrases they were given showed a number of common points. The common methods used to create abbreviations were the first four of those listed above. Even the shortest words usually had more than one abbreviation (mean of 3.35 in Hodge et al. and 5.73 in Streeter et al.), with the average number of abbreviations per word increasing with word length (mean of 6.0 in Hodge et al. and 18.0 in Streeter et al.). The most common algorithm used for shorter words was vowel deletion, while longer words tended to be truncated. Streeter et al. point out that vowel deletion requires producing the whole word and then deleting the vowels, an effort-prone and time-consuming task for long words (small words probably being handled as a single *chunk*). In the case of polysyllabic words truncation produces short abbreviations, which are easy to produce and only require a single change to the word output strategy (cut it short).

- A study by Hodge and Pennington[143] asked subjects to create a personal (one they might use for their own private writing) and a general (one that could be understood by other people) abbreviation from words containing between four and nine letters. The results for the personal abbreviations paralleled those of the general abbreviations. Male subjects made greater use of vowel removal for the longer words than female subjects (who preferred truncation for longer words). The percentage of the original word's letters used in the abbreviation decreased with word length (from 70–75% for shorter words to 58–65% for longer words). The abbreviations of more frequent words contained [782] word frequency fewer letters than less frequent words. A separate group of subjects were given the abbreviations created by the first group and asked to reconstruct the original words. The mean reconstruction rate at all word lengths was 67%.
- A study by Streeter, Ackroff, and Taylor[297] investigated the rules used by people to create abbreviations. Subjects were asked to produce "good abbreviations" for 81 computer command names and arguments (e.g., *move* and "usage billing number"). Analysis of the results was based on the number of syllables in a word (between one and four) or the input containing multiple words. The resulting abbreviations were compared against those produced by a variety of algorithms. The performance of the different algorithms varied with the number of syllables (Figure 33).

  A second group of subjects were asked to learn word/abbreviation pairs. The abbreviations used were those generated by the first group of subjects. The abbreviations used were either the most popular one chosen for a word, or the one obtainable by following the Streeter et al. rules. When given the word and asked for the abbreviation, mean recall performance was 54% correct for the popular abbreviations and 70% for the rule abbreviations (recall rate decreased in both cases as the number of syllables increased, although both were more than 90% for multiple words). Another two experiments, using randomly chosen English words, paralleled the procedure in the first two experiments.

However, after learning word/abbreviation pairs, subjects were asked to recall the word when given the abbreviation (mean recall performance was 62.6% correct for the popular abbreviations and 46.7% for the rule abbreviations, with recall rate slowly decreasing in both cases as the number of syllables increased).

An alternative approach to predicting use of abbreviation strategies was studied by Carter and Clopper.[51] Words are abbreviated in both spoken as well as written forms—for instance, *rhinoceros* ⇒ *rhino* and *telephone* ⇒ *phone*. Subjects were asked to listen to a series of words. After each word, they had to speak the word and then produce a *reduced* spoken form (they were reminded that most of the words would not normally be reduced in everyday speech).

Table 21: The syllable most likely to be omitted in a word (indicated by the × symbol) based on the number of syllables (*syl*) and the position of the primary, (*pri*) stressed syllable. Adapted from Carter and Clopper.[51]

Syllables in Word and Primary Stress Position	Syllable(s) Omitted 1	2	Most 3	Often 4
2syl–1pri		×	–	–
2syl–2pri	×		–	–
3syl–1pri		×	×	–
3syl–2pri	×			–
3syl–3pri		×	×	–
4syl–1pri		×		
4syl–2pri			×	×
4syl–3pri	×	×		×

Carter and Clopper drew three conclusions from the results (Table 21):

1. the stressed syllable is nearly always preserved,
2. the initial syllable is preserved more often than omitted, and
3. only when the final syllable of a two syllable word contains the stress is that syllable preserved more often than it is omitted.

A study by Bourne and Ford[34] investigated word abbreviation from an information content point of view. They looked at thirteen different algorithms capable of reducing an arbitrary word to a predefined number of letters. Algorithm quality was measured by the ability to map different words to different abbreviations. The consistently best algorithm dropped every second letter (this rule was applied iteratively on the successively shorter letter sequences until the desired number of characters was obtained) and appended a check letter (algorithmicly derived from the discarded letters). While this algorithm might be of interest when automatically generating identifier spellings, we are only interested in human-created spellings here.

The following studies investigated the interpretation of existing abbreviations in various contexts:

- Sproat, Black, Chen, Kumar, Ostendorf, and Richards[288] give a detailed discussion of the issues involved in converting what they call *nonstandard* words to spoken words (in the context of text to speech synthesis). What appears to be an abbreviation may actually be a nonstandard word. For instance, `Article_IV` is probably pronounced "article four", `Henry_IV` is pronounced as "Henry the fourth", while `IV_drip` is probably pronounced "I V drip".
- Laitinen, Taramaa, Heikkilä, and Rowe[188] built a tool, `InName`, to *disabbreviate* source code. `InName` used a simple grammar to describe the components of an identifier spelling. This broke the spelling into short letter sequences which were assumed to be either abbreviations of words or words (e.g., `boOffsetMeasDone` ⇒ `bo`, `Offset`, `Meas`, `Done`). A relatively small dictionary of around 1,000 entries was used to detect words, plus a list of 300 common abbreviations (e.g., len ⇒ length, curr ⇒

current). A GUI interface highlighted an abbreviated name and listed possible nonabbreviated forms (e.g., `tmpnamelen` ⇒ `temporary_name_length`). The user could accept one of the suggested forms or type in their own choice. The results of not abbreviating five applications are shown in Table 22.

Table 22: Five different applications (A–E) unabbreviated using `InName`, by five different people. Application C had many short names of the form `i`, `m`, `k`, and `r2`. Adapted from Laitinen.[188]

Application	A	B	C	D	E
Source lines	12,075	6,114	3,874	6,420	3,331
Total names	1,410	927	439	740	272
Already acceptable	5.6	3.1	8.7	9.3	11.0
Tool suggestion used	42.6	44.7	35.3	46.8	41.5
User suggestion used	39.6	29.3	15.0	30.7	43.8
Skipped or unknown names	12.2	22.9	41.0	13.2	3.7
User time (hours)	11	5	4	4	3

- A study by Anquetil and Lethbridge[11] investigated the abbreviations used to name files (this is discussed elsewhere). [] file name abbreviations

6.7 Implementation and maintenance costs

Encoding information in the sequence of characters forming an identifier's spelling sounds attractive and this usage often occurs in automatically generated code. However, human-written code is invariably maintained by humans and a number of possible human-related factors ought to be taken into account. While it might not yet be possible to obtain reliable figures on these costs, the main ones are listed here for future reference, including:

- Keeping the information up-to-date during source code maintenance and updates.
- The cost of new readers learning to interpret the encoding used.
- The probability of correctly interpreting the character sequences used. For instance, a numeric value may indicate some maximum value, but this maximum may be the largest representable value (a representation attribute) or the largest value an object is expected to hold (an application attribute).
- The cost of processing the character encoding when reading source. (It took developers many years of practice to achieve fluency in reading text in their native language and they are likely to require practice at decoding identifier spellings before they can read them as fluently.)

The only information likely to be needed every time the identifier is read is the semantics of what it denotes. Experience suggests that this information rarely changes during the development and maintenance of a program.

6.8 Typing mistakes

typing mistakes

A summary of the results of typing studies would include two factors of relevance to this identifier guidelines section. Hitting a key adjacent to the correct one is the largest single contributor (around 35%) to the number of typing mistakes made by people. Performance is affected by the characteristics of the typists native written language (word frequency and morphology). [782] morphology

Researchers studying typing often use skilled typists as subjects (and build models that mimic such people[268]). These subjects are usually asked to make a typewritten copy of various forms of prose—the kind of task frequently performed by professional typists, the time taken and errors made being measured. Software developers are rarely skilled typists and rarely copy from written material. (It is often created in the developer's head on the fly, and a theory of developer typing performance would probably need to consider these two processes separately.)

These coding guidelines assume that developers' typing mistakes will follow the same pattern as those of typists, although the level of performance may be lower. It is also assumed that the primary input device will be a regular-size keyboard and not one of those found on mobile computers.[206]

- A study by Shaffer and Hardwick[275] asked qualified touch typists to type text, the characteristics of which varied. The five different kinds of text were: prose, an article on gardening; word, a random arrangement of the words in the previous article; syllable, obtained by shuffling the spaces between the words of the article into syllable boundaries within the words; first-order, random letter strings having the same distribution of letters as the article; and zero-order, random letter strings with all letters being equally probable.

Table 23: Distribution of mistakes for each kind of text. Unparenthesized values are for subjects making fewer than 2.5% mistakes, and parenthesized values for subjects making 2.5% or more mistakes. Omission—failing to type a letter; response—hitting a key adjacent to the correct one; reading—mistakes were those letters that are confusable visually or acoustically; context—transpositions of adjacent letters and displacements of letters appearing within a range of three letters left or right of the mistake position; random—everything else. When a mistake could be assigned to more than one category, the category appearing nearer the top of the table was chosen. Adapted from Shaffer.[275]

Kind of mistake	Prose	Word	Syllable	First Order	Zero Order	Total
Omission	19 (21)	11 (23)	24 (36)	15 (46)	34 (82)	103 (208)
Response	19 (25)	31 (38)	27 (53)	32 (43)	108 (113)	217 (272)
Reading	3 (2)	2 (0)	8 (15)	14 (20)	20 (41)	47 (78)
Context	19 (27)	19 (17)	34 (30)	56 (51)	46 (40)	174 (165)
Random	3 (5)	2 (6)	4 (11)	13 (15)	22 (41)	44 (78)
Total	63 (80)	65 (84)	97 (145)	130 (175)	230 (317)	585 (801)

The results (Table 23) show that hitting a key adjacent to the correct one was the largest single contributor (around 35%) to the number of mistakes. Surprisingly, both the number of mistakes and typing rate were the same for prose and random word ordering. Text containing words created using purely random letter sequences had the highest rate of typing mistakes (and the slowest typing rate), almost twice that of text created using the distribution of letters found in English. Shaffer and Hardwick performed a second experiment to investigate the reasons for the significant difference in typing performance when first- or zero-order words were used. Was it caused by a decline in syllable-like sequences in the words because of fewer vowels, or because of an increase in the less frequently used letters of the alphabet? Every letter of the alphabet was used 10 times to create passages of 52 five-letter words and 16 fifteen-letter words (plus a single 20-letter word). For one set of passages the letters in each word were randomly selected; in the other set an attempt was made to create words containing readable syllables (e.g., *spowd*, *throx*).

Table 24: Mean response time per letter (in milliseconds). Right half of the table shows mean response times for the same subjects with comparable passages in the first experiment. Adapted from Shaffer.[275]

	Syllable	Random		First Order	Zero Order
5-letter	246	326	Fixed	236	344
15-letter	292	373	Random	242	343

The only results reported (Table 24) were response times for letters typed, not the number of mistakes. These results show that performance for text containing words having readable syllables was significantly better than words having a random sequence of letters. Since both passages contained the same number of occurrences of each letter, the difference was not caused by a decrease in the number of vowels or an increase in the number of infrequently used letters. Performance was slower for the passage containing longer words.

- A study by Gentner, Larochelle, and Grudin[117] also found that rate of typing was affected by letter digraph frequency and word frequency in the typist's natural language (they did not measure error mistake rates). The position of the digraph within the word and syllable boundaries had a smaller affect on performance.
- A study by Schoonard and Boies[271] taught subjects to use abbreviations for commonly occurring words (so-called *short-type*; the intent being to increase typing performance by having the word processor used automatically expand the abbreviations). The results showed an average short-type detection rate of 93.2% (of those possible) and that typing rate (in characters per second) was not affected by use of short-type (error rates were only given for short-type). Developers often abbreviate words when creating identifier names (but editors rarely expand them).

  [] people error rates
  [] abbreviating identifier
- Studies have analyzed the mistakes made by each finger of each hand. Software developers rarely touch type, often using a few fingers from each hand. *Stewardesses* may be the longest English word touch typed with the left hand only, but most developers also use fingers from the right hand. For this reason these studies are not considered applicable here.

6.9 Usability of identifier spelling recommendations

identifier guideline significant characters

For programs containing large numbers of identifiers the computational resources required to enforce a recommendation that identifiers differ by some minimum amount may have significant usability implications (e.g., developers having to wait for a relatively long period of time for a proposed choice of identifier spelling to be checked against existing identifier spellings). One way of reducing the computational resources required is to reduce the number of identifiers that need to be checked. All identifiers have attributes other than their spelling (they all exist in some name space and scope, have a linkage and some have a type), and it might be possible to take advantage of the consequences of an identifier having these attributes; for instance:

[782] Levenstein distance

[] name space
[] scope
[] linkage
[] types

- If the identifier `X_1` is not visible at the point in the source where a developer declares and uses the identifier `X_2`, a reference to `X_2` mistyped as `X_1` will result in a translator diagnostic being issued.

  [] visible identifier
- If the identifiers `X_1` and `X_2` are in different name spaces, a mistyped reference to one can never result in a reference to the other.

  [] name space
- If the identifiers `X_1` and `X_2` are objects or functions having incompatible types, a mistyped reference to one, which refers to the other, is likely to result in a translator diagnostic being issued.

Mistyping an identifier only becomes a fault if the usage does not cause a diagnostic to be issued by a translator. (A developer who visually confuses two identifiers can create a fault that does not generate a translator diagnostic by writing code that makes use of incorrect identifier information without directly referencing the confused identifiers.)

While modifications to existing source may not result in new identifiers being declared, it is possible for the C language attributes of an existing identifier to be changed. Experience suggests that modifications rarely change an identifier's name space, but that changes of scope or linkage (which controls visibility) is relatively common; for instance, moving the declaration of an identifier from block scope to file scope or changing its linkage from internal to external (experience suggests that these changes rarely occur in the other direction). Changes to the type of an object might include a change of integer type or new members being added to the structure type it has.

[] scope overlapping

This usage pattern suggests the following deviation:

DEV Every identifier need only be compared against every other identifier in the same name space in the visible source of a program.

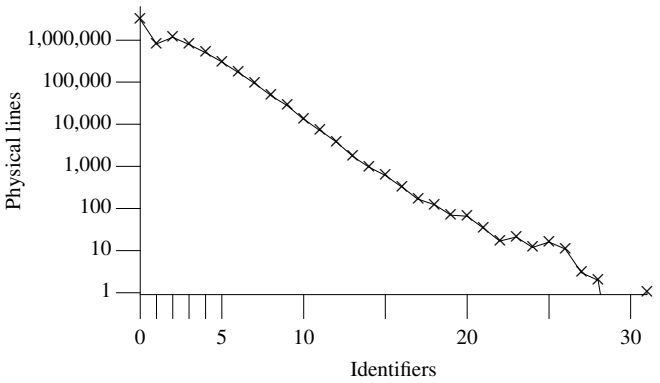


Figure 34: Number of physical lines containing the given number of identifiers, based on the visible form of the `.c` files.

Usage

References

At the end of each entry, the pages on which that entry is cited are listed in parentheses.

[1] P. A. Adams and J. K. Adams. Confidence in the recognition and reproduction of words difficult to spell. *American Journal of Psychology*, 73:544–552, 1960.

[2] J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[3] P. A. Allen, B. Wallace, and T. A. Weber. Influence of case type, word frequency, and exposure duration on visual word recognition. *Journal of Experimental Psychology: Human Perception and Performance*, 21(4):914–934, 1995.

[4] C. J. Álvarez, M. Carreiras, and M. de Vega. Syllable-frequency effect in visual word recognition: Evidence of sequential-type processing. *Psicológica*, 21:341–374, 2000.

[5] L. A. N. Amaral, A. Scala, M. Barthélémy, and H. E. Stanley. Classes of small-world networks. *Proceedings of the National Academy of Sciences*, 97(21):11149–11152, Oct. 2000.

[6] J. R. Anderson. *Learning and Memory*. John Wiley & Sons, Inc, second edition, 2000.

[7] S. Andrews. Lexical retrieval and selection processes: Effects of transposed-letter confusability. *Journal of Memory and Language*, 35:775–800, 1996.

[8] S. Andrews. The effect of orthographic similarity on lexical retrieval: Resolving neighborhood conflicts. *Psychonomic Bulletin & Review*, 4(4):439–461, 1997.

[9] S. Andrews and D. R. Scarratt. Rule and analogy mechanisms in reading nonwords: Hough dou peapel rede gnew wirds? *Journal of Experimental Psychology: Human Perception and Performance*, 24(4):1052–1086, 1998.

[10] Ángel Fernández and M. A. Alonso. The relative value of environmental context reinstatement in free recall. *Psicológica*, 22:253–266, 2001.

[11] N. Anquetil and T. Lethbridge. Extracting concepts from file names; a new file clustering criterion. In *Proceedings of the 1998 International Conference on Software Engineering*, pages 84–93. IEEE Computer Society Press / ACM Press, 1998.

[12] A. D. Baddeley. How does acoustic similarity influence short-term memory? *Quarterly Journal of Experimental Psychology*, 20:249–264, 1968.

[13] A. D. Baddeley. Language habits, acoustic confusability, and immediate memory for redundant letter sequences. *Psychonomic Science*, 22(2):120–121, 1971.

[14] A. D. Baddeley. *Essentials of Human Memory*. Psychology Press, 1999.

[15] P. Baggett and A. Ehrenfeucht. How an unfamiliar thing should be called. *Journal of Psycholinguistic Research*, 11(5):437–445, 1982.

[16] J. R. Barclay, J. D. Bransford, J. J. Franks, N. S. McCarrell, and K. Nitsch. Comprehension and semantic flexibility. *Journal of Verbal Learning and Verbal Behavior*, 13:471–481, 1974.

[17] C. Barry and P. H. K. Seymour. Lexical priming and sound-to-spelling contingency effects in nonword spelling. *The Quarterly Journal of Experimental Psychology*, 40A(1):5–40, 1988.

[18] L. Bauer. *English Word-formation*. Cambridge University Press, 1983.

[19] L. Bauer. *An Introduction to International Varieties of English*. Edinburgh University Press, 2002.

[20] J. P. Beldie, S. Pastoor, and E. Schwarz. Fixed versus variable letter width for television text. *Human Factors*, 25(3):273–277, 1983.

[21] B. Berlin and P. Kay. *Basic Color Terms*. Berkeley: University of California Press, 1969.

[22] R. S. Berndt, J. A. Reggia, and C. C. Mitchum. Empirical derived probabilities for grapheme-to-phoneme correspondences in English. *Behavior and Research Methods, Instruments, & Computers*, 19(1):1–9, 1987.

[23] R.-J. Beun and A. H. M. Cremers. Multimodal reference to objects: An empirical approach. In H. Bunt and R.-J. Beun, editors, *Cooperative Multimodal Communication*, pages 64–88. Springer-Verlag, 1998.

[24] D. Biber, S. Johansson, G. Leech, S. Conrad, and E. Finegan. *Longman Grammar of Spoken and Written English*. Pearson Education, 1999.

[25] P. P. G. Boersma. *Functional Phonology*. PhD thesis, University of Amsterdam, Sept. 1998.

[26] C. Boiarsky. Consistency of spelling and pronunciation deviations of appalachian students. *Modern Language Journal*, 53:347–350, 1969.

[27] D. B. Boles and J. E. Clifford. An upper- and lowercase alphabetic similarity matrix, with derived generation similarity values. *Behavior Research Methods, Instruments, & Computers*, 21(6):579–586, 1989.

[28] L. Boroditsky. Metaphoric structuring: understanding time through spatial metaphors. *Cognition*, 75:1–28, 2000.

[29] L. Boroditsky. Does language shape thought?: Mandarin and English speaker' conception of time. *Cognitive Psychology*, 43:1–22, 2001.

[30] P. Boucher. Teaching compound nouns in English: an application of the Charlie Brown principle. *CIEREC/TRAVAUX LXXVI*, pages 33–50, 1992.

[31] P. Boucher, F. Danna, and P. Sébillot. Compounds: An intelligent tutoring system for learning to use compounds in English. Technical Report Publication Interne No. 718, IRISA, Campus Universitaire de Beaulieu, France, Apr. 1993.

[32] H. Bouma. Visual recognition of isolated lower-case letters. *Vision Research*, 11:459–474, 1971.

[33] C. P. Bourne. Frequency and impact of spelling errors in bibliographic data bases. *Information Processing & Management*, 13(1):1–12, 1997.

[34] C. P. Bourne and D. F. Ford. A study of methods for systematically abbreviating English words and names. *Journal of the ACM*, 8(4):538–552, 1961.

[35] J. D. Bransford and M. K. Johnson. Contextual prerequisites for understanding: Some investigations of comprehension and recall. *Journal of Verbal Learning and Verbal Behavior*, 11:717–726, 1972.

[36] S. Brédart. Retrieval failure in face naming. *Memory*, 1:351–366, 1993.

[37] T. Brennen. The difficulty with recalling people's names: The plausible phonology hypothesis. *Memory*, 1(4):409–431, 1993.

[38] L. Brooks. Visual pattern in fluent word identification. In A. S. Reber and D. L. Scarborough, editors, *Toward a Psychology of Reading: The Proceedings of the CUNY Conference*, chapter 3, pages 143–181. Erlbaum, 1977.

[39] J. O. Brooks III, L. Friedman, J. M. Gibson, and J. A. Yesavage. Spontaneous mnemonic strategies used by older and younger adults to remember proper names. *Memory*, 1(4):393–407, 1993.

[40] H. D. Brown. Categories of spelling difficulty in speakers of English as a first and second language. *Journal of Verbal Learning and Verbal Behavior*, 9:232–236, 1970.

[41] J. Brown, V. J. Lewis, and A. F. Monk. Memorability, word frequency and negative recognition. *Quarterly Journal of Experimental Psychology*, 29:461–473, 1977.

[42] R. W. Brown, A. H. Black, and A. E. Horowitz. Phonetic symbolism in natural languages. *Journal of Abnormal and Social Psychology*, 50:388–393, 1955.

[43] L. Buchanan, N. R. Brown, R. Cabeza, and C. Maitson. False memories and semantic lexicon arrangement. *Brian and Language*, 68:172–177, 1999.

[44] A. Budanitsky. Lexical semantic relatedness and its application in natural language processing. Technical Report CSRG-390, Computer Systems Research Group, University of Toronto, Aug. 1999.

[45] H. Bunke. A fast algorithm for finding the nearest neighbor of a word in a dictionary. Technical Report IAM-93-025, Institut für Informatik, Nov. 1993.

[46] C. Burgess and K. Livesay. The effect of corpus size on predicting reaction time in a basic word recognition task: Moving on from Kučera and Francis. *Behavior Research Methods, Instruments, & Computers*, 30(2):272–277, 1998.

[47] D. M. Burke, L. Peters, and R. M. Harrold. Word association norms for young and older adults. *Social and Behavioral Science Documents*, 17(2):???, 1987.

[48] B. Caprile and P. Tonella. Nomen est omen: Analyzing the language of function identifiers. In *Proceedings of WCRE'99, Working Conference on Reverse Engineering*, pages 112–122, Oct. 1999.

[49] M. Carlo and E. S. Sylvester. Adult second-language reading research: How may it inform assessment and instruction? Technical Report TR96-08, University of Pennsylvania, Oct. 1996.

[50] J. M. Carroll. *What's in a Name? An essay on the psychology of reference*. W. H. Freeman, 1985.

[51] A. K. Carter and C. G. Clopper. Prosodic and morphological effects on word reduction in adults: A first report. Technical Report Progress Report No. 24 (2000), Speech Research Laboratory, Indiana University, 2000.

[52] M. Celce-Murcia, D. M. Brinton, and J. M. Goodwin. *Teaching Pronunciation: A Reference for Teachers of English to Speakers of Other Languages*. Cambridge University Press, 1996.

[53] M. Celce-Murcia and D. Larsen-Freeman. *The Grammar Book: An ESL/EFL Teacher's Course*. Heinle & Heinle, second edition, 1999.

[54] S. M. Chambers and K. I. Forster. Evidence for lexical access in a simultaneous matching task. *Memory & Cognition*, 3(5):549–559, 1975.

[55] H.-F. Chitiri and D. M. Willows. Word recognition in two languages and orthographies: English and Greek. *Memory & Cognition*, 22(3):313–325, 1994.

[56] P. Christian. Soundex - can it be improved? *Computers in Geneology*, 6(5):???, 1998.

[57] H. H. Clark. *Understanding language*. Cambridge University Press, 1996.

[58] H. H. Clark and D. Wilkes-Gibbs. Referring as a collaborative process. *Cognition*, 22:1–39, 1986.

[59] G. Cohen. Why is it difficult to put names to faces? *British Journal of Psychology*, 81:287–297, 1990.

[60] G. Cohen and D. Faulkner. Memory for proper names: Age differences in retrieval. *British Journal of Psychology*, 4:187–197, 1986.

[61] J. Coleman and J. Pierrehumbert. Stochastic phonological grammars and acceptability. In *Computational phonology: Third meeting of the ACL special interest group in computational phonology*, pages 49–56. Association for the Computational Linguistics, 1997.

[62] Collins. *Advanced Learner's English Dictionary*. Collins COBUILD, 2003.

[63] M. Coltheart, K. Rastle, C. Perry, R. Langdon, and J. Ziegler. DRC: A dual route cascaded model of visual word recognition and reading aloud. *Psychological Review*, 108(1):204–256, 2001.

[64] V. Coltheart. Effects of phonological similarity and concurrent irrelevant articulation on short-term-memory recall of repeated and novel word lists. *Memory & Cognition*, 21(4):539–545, 1993.

[65] B. Comrie. *Language Universals and Linguistic Typology*. Blackwell, second edition, 1989.

[66] R. Conrad. Acoustic confusions in immediate memory. *British Journal of Psychology*, 55(1):75–84, 1964.

[67] R. Conrad. Order error in immediate recall of sequences. *Journal of Verbal Learning and Verbal Behavior*, 4:161–169, 1965.

[68] V. Cook. *Second Language Learning and Language Teaching*. Arnold, third edition, 2001.

[69] V. J. Cook. *Inside Language*. Arnold, 1997.

[70] V. J. Cook. L2 users and English spelling. *Journal of Multilingual and Multicultural Development*, 18(6):474–488, 1997.

[71] R. Coppieters. Competence differences between native and near-native speakers. *Language*, 63(3):544–573, 1987.

[72] G. G. Corbett and I. R. L. Davies. Establishing basic color terms: measures and techniques. In C. L. Hardin and L. Maffi, editors, *Color categories in thought and language*, chapter 9, pages 197–223. Cambridge University Press, 1997.

[73] D. W. J. Corcoran. Acoustic factor in proof reading. *Nature*, 214:851–852, May 1967.

[74] F. Costello and M. T. Keane. Polysemy in conceptual combination: Testing the constraint theory of combination. In P. L. M. G. Shafto, editor, *Nineteenth Annual Conference of the Cognitive Science Society*, pages 137–142. Erlbaum, Apr. 1997.

[75] F. J. Costello. Efficient creativity: Constraint-guided conceptual combination. *Cognitive Science*, 24(2):299–349, 2000.

[76] F. J. Costello. Testing a computational model of categorisation and category combination: Identifying disease categories and new disease combinations. In *Proceedings of Twenty-Third Annual Conference of the Cognitive Science Society*, page ???, ???, Aug. 2001. ???

[77] F. J. Costello and M. T. Keane. Testing two theories of conceptual combination: Alignment versus diagnosticity in the comprehension and production of combined concepts. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 27(1):255–271, 2001.

[78] A. Cruttenden. *Gimson's Pronunciation of English*. Arnold, sixth edition, 2001.

[79] A. Cutler. The reliability of speech error data. In A. Cutler, editor, *Slips of the tongue and language production*, chapter Guest editorial, pages 7–28. Mouton, 1982.

[80] W. Daelemans and A. van den Bosch. Language-independent data-oriented grapheme-to-phoneme conversion. In J. P. H. van Santen, R. W. Sproat, J. P. Olive, and J. Hirschberg, editors, *Progress in Speech Synthesis*, pages 77–89. Springer, New York, 1997.

[81] B. Daille, B. Habert, C. Jacquemin, and J. Royauté. Empirical observation of term variations and principles for their description. *Terminology*, 3(2):197–258, May 1996.

[82] R. Dale and E. Reiter. Computational interpretations of the Gricean maxims in the generating referring expression. *Cognitive Science*, 19(2):233–263, 1995.

[83] F. J. Damerau and E. Mays. An examination of undetected typing errors. *Information Processing & Management*, 25(6):659–664, 1989.

[84] R. I. Damper, Y. Marchand, M. J. Adamson, and K. Gustafson. Evaluating the pronunciation component of text-to-speech systems for English: A performance comparison of different approaches. *Computer Speech and Language*, 13(2):155–176, 1999.

[85] M. Daneman and M. Stainton. Phonological recoding in silent reading. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 17(4):618–632, 1991.

[86] P. Desberg, D. E. Elliott, and G. Marsh. American black English spelling. In U. Frith, editor, *Cognitive Processes in Spelling*, chapter 4, pages 69–82. Academic Press, 1980.

[87] N. Deshmukh. *Maximum likelihood estimation of multiple pronunciations for proper names*. PhD thesis, Mississippi State University, June 1999.

[88] C. DiMarco, G. Hirst, and M. Stede. The semantic and stylistic differentiation of synonyms and near-synonyms. In *AAAI Spring Symposium on Building Lexicons for Machine Translation*, pages 114–121, Mar. 1993.

[89] R. Dirven. Dividing up physical and mental space into conceptual categories by means of English prepositions. In C. Zelinsky-Wibbelt, editor, *Natural Language processing (vol. 3, The Semantics of Prepositions)*, pages 73–97. Mouton de Gruyter, 1993.

[90] M. Divay and T. Vitale. Algorithms for grapheme-phoneme translation for English and French: Applications for database searches and speech synthesis. *Computational Linguistics*, 23(4):496–523, 1997.

[91] P. Downing. On the creation and use of English compound nouns. *Language*, 53(4):810–842, 1977.

[92] A. Drewnowski and A. F. Healy. Phonetic factors in letter detection: A reevaluation. *Memory & Cognition*, 10(2):145–154, 1982.

[93] P. Edmonds. *Semantic Representation of Near-Synonyms for Automatic Lexical Choice*. PhD thesis, University of Toronto, 1999.

[94] S. L. Ehrenreich and T. Porcu. Abbreviations for automated systems: Teaching operators the rules. In A. Badre and B. Shneiderman, editors, *Directions in Human/Computer Interaction*, chapter 6, pages 111–135. Ablex Publishing Corp., 1982.

[95] W. H. Eichelman. Familiarity effects in the simultaneous matching task. *Journal of Experimental Psychology*, 86(2):275–282, 1970.

[96] J. Ellman. *Using Roget's Thesaurus to Determine Similarity of Texts*. PhD thesis, University of Sunderland, June 2000.

[97] S. Farrell and S. Lewandowsky. Dissimilar items benefit from phonological similarity in serial recall: The dissimilar immunity effect reviseted. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 29(5):838–849, 2003.

[98] D. Fay and A. Cutler. Malapropisms and the structure of the mental lexicon. *Linguistic Inquiry*, 8(3):505–520, 1977.

[99] C. Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

[100] C. J. Fillmore. Topics in lexical semantics. In R. W. Cole, editor, *Current Issues in Linguistic Theory*, pages 76–138. Indiana University Press, 1977.

[101] J. H. Flowers and D. J. Lohr. How does familiarity affect visual search for letter strings? *Perception and Psychophysics*, 37:557–567, 1985.

[102] J. R. Folk. Phonological codes are used to access the lexicon during silent reading. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 25(4):892–906, 1999.

[103] U. H. Frauenfelder, R. H. Baayen, F. M. Hellwig, and R. Schreuder. Neighborhood density and frequency across languages and modalities. *Journal of Memory and Language*, 32:781–804, 1993.

[104] S. Frisch. *Similarity and frequency in phonology*. PhD thesis, Northwestern University, Dec. 1996.

[105] S. A. Frisch, N. R. Large, and D. B. Pisoni. Perception of wordlikeness: Effects of segment probability and length of the processing of nonwords. *Journal of Memory and Language*, 42:481–496, 2000.

[106] R. Frost. Phonological computations and missing vowels: Mapping lexical involvement in reading. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 21(2):398–408, 1995.

[107] R. Frost, L. Katz, and S. Bentin. Strategies for visual word recognition and orthographical depth: A multilingual comparison. *Journal of Experimental Psychology: Human Perception and Performance*, 13(1):104–115, 1987.

[108] E. Fudge. *English Word-Stress*. George Allen & Unwin, 1984.

[109] M. Fuller and J. Zobel. Conflation-based comparison of stemming algorithms. In *Proceedings of the Third Australian Document Computing Symposium*, pages 8–13, Aug. 1998.

[110] G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais. Statistical semantics: Analysis of the potential performance of key-word information systems. *The Bell System Technical Journal*, 62(6):1753–1805, 1983.

[111] G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais. The vocabulary problem in human-system communication: an analysis and a solution. *Communications of the ACM*, 30(11):964–971, 1987.

[112] T. Furugori. Improving spelling checkers for Japanese users of English. *IEEE Transactions on Professional Communication*, 33(3):138–142, Sept. 1990.

[113] X. Gabaix. Zipf's law for cities: An explanation. *The Quarterly Journal of Economics*, 114(4):739–767, Aug. 1999.

[114] R. Ganesan and A. T. Sherman. Statistical techniques for language recognition: An introduction and guide for cryptanalysts. *Cryptologia*, XVII(4):321–366, Oct. 1993.

[115] M. K. Gardner, E. Z. Rothkopf, R. Lapan, and T. Laferty. The word frequency effect in lexical decision: Finding a frequency-based component. *Memory & Cognition*, 15(1):24–28, 1987.

[116] S. Garrod and G. Doherty. Conversation, co-ordination and convention: an empirical investigation of how groups establish linguistic conventions. *Cognition*, 53:181–215, 1994.

[117] D. R. Gentner, S. Larochelle, and J. Grudin. Lexical, sublexical, and peripheral effects in skilled typewriting. *Cognitive Psychology*, 20:524–548, 1988.

[118] M. J. Gervais, L. O. Harvey, and J. O. Roberts. Identification confusions among letters of the alphabet. *Journal of Experimental Psychology: Human Perception and Performance*, 10(5):655–666, 1984.

[119] E. J. Gibson, A. Pick, H. Osser, and M. Hammond. The role of grapheme-phoneme correspondence in the perception of words. *American Journal of Psychology*, 75:554–570, 1962.

[120] G. C. Gilmore, H. Hersh, A. Caramazza, and J. Griffin. Multidimensional letter similarity derived from recognition errors. *Perception & Psychophysics*, 25(5):425–431, 1979.

[121] D. R. Godden and A. D. Baddeley. Context-dependent memory in two natural environments: On land and underwater. *British Journal of Psychology*, 66(3):325–331, 1975.

[122] A. R. Golding and P. S. Rosenbloom. A comparison of Anapron with seven other name-pronunciation systems. Technical Report TR-93-05a, Mitsubishi Electric Research Laboratories, May 1996.

[123] J. M. Goldschneider and R. M. DeKeyser. Explaining the "Natural order of L2 morpheme acquisition" in English: A meta-analysis of multiple determinants. *Language Learning*, 51(1):1–50, 2001.

[124] P. F. D. Gontijo, J. Rayman, S. Zhang, and E. Zaidel. How brand names are special: brands, words, and hemispheres. *Brain and Language*, 82:327–343, 2002.

[125] H. Goodglass and A. Wingfield. Selective preservation of a lexical category in aphasia: Disassociations in comprehension of body parts and geographical places names following focal brain lesion. *Memory*, 1(4):313–328, 1993.

[126] J. Grainger, J. K. O'Regan, A. M. Jacobs, and J. Segui. Neighborhood frequency effects and letter visibility in visual word recognition. *Perception & Psychophysics*, 51(1):49–56, 1992.

[127] D. Green and P. Meara. The effects of script on visual search. *Second Language Research*, 3(2):102–118, 1987.

[128] D. W. Green, E. J. Hammond, and S. Supramaniam. Letters and shapes: Developmental changes in search strategies. *British Journal of Psychology*, 74:11–16, 1983.

[129] P. Grice. *Studies in the Way of Words*. Harvard University Press, 1989.

[130] H. Haarmann and M. Usher. Maintenance of semantic information in capacity-limited short-term memory. *Psychonomic Bulletin & Review*, 8(3):568–578, 2001.

[131] R. N. Haber and R. M. Schindler. Error in proofreading: Evidence of syntactic control of letter processing? *Journal of Experimental Psychology: Human Perception and Performance*, 7(3):573–579, 1981.

[132] J. R. Hanley and P. Morris. The effects of amount of processing on recall and recognition. *Quarterly Journal of Experimental Psychology*, 39A:431–449, 1987.

[133] C. L. Hardin and L. Maffi. *Color categories in thought and language*. Cambridge University Press, 1997.

[134] T. Harley. *The Psychology of Language*. Psychology Press, second edition, 2001.

[135] M. W. Harm. *Division of Labor in a Computational Model of Visual Word Recognition*. PhD thesis, University of Southern California, Aug. 1998.

[136] J. A. Hawkins and G. Gilligan. Prefixing and suffixing universals in relation to basic word order. *Lingua*, 74:219–258, 1988.

[137] A. F. Healy and T. F. Cunningham. A developmental evaluation of the role of word shape in word recognition. *Memory & Cognition*, 20(2):141–150, 1992.

[138] L. Henderson and S. E. Henderson. Visual comparison of words and random letter strings: Effects of number and position of letters different. *Memory & Cognition*, 3(1):97–101, 1975.

[139] S. Henser. Thinking in Japanese? What have we learned about language-specific thought since Ervin Tripp's psychological tests of Japanese-English bilinguals. Technical Report No. 32, Nissan Institute of Japanese Studies, 2000.

[140] R. N. A. Henson. Item repetition in short-term memory: Ranschburg repeated. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 24(5):1162–1181, 1998.

[141] C. M. Herdman, D. Chernecki, and D. Norris. Naming cAsE aLtErNaTeD words. *Memory & Cognition*, 27(2):254–266, 1999.

[142] C. M. Herdman and A. R. Dobbs. Attentional demands of visual word recognition. *Journal of Experimental Psychology: Human Perception and Performance*, 15(1):124–132, 1989.

[143] M. H. Hodge and F. M. Pennington. Some studies of word abbreviation behavior. *Journal of Experimental Psychology*, 98(2):350–361, 1973.

[144] M. B. Holbrook. A comparison of methods for measuring the interletter similarity between capital letters. *Perception & Psychophysics*, 17(6):532–536, 1975.

[145] M. B. Holbrook. Effect of subjective interletter similarity, perceived word similarity, and contextual variables on the recognition of letter substitutions in a proofreading task. *Perceptual and Motor Skills*, 47:251–258, 1978.

[146] J. Hollis and T. Valentine. Proper name processing: Are proper names pure referencing expressions? *Journal of Experimental Psychology: Learning, Memory & Cognition*, 27:99–116, 2001.

[147] A. Holm and B. Dodd. The effect of first written language on the acquisition of English literacy. *Cognition*, 59:119–147, 1996.

[148] D. Holmes and M. C. McCabe. Improving precision and recall for soundex retrieval. In *Proceedings of the 2002 IEEE International Conference on Information Technology - Coding and Computing (ITCC)*, pages 22–27, Apr. 2002.

[149] L. M. Horowitz. Free recall and ordering of trigrams. *Journal of Experimental Psychology*, 62(1):51–57, 1961.

[150] D. H. Howes and R. L. Solomon. Visual duration threshold as a function of word-probability. *Journal of Experimental Psychology*, 41:401–410, 1951.

[151] C.-W. Hue and J. R. Erickson. Short-term memory for Chinese characters and radicals. *Memory & Cognition*, 16(3):196–205, 1988.

[152] A. J. Hull. A letter-digit metric of auditory confusions. *British Journal of Psychology*, 64(4):579–585, 1973.

[153] D. A. Hull and G. Grefenstette. A detailed analysis of English stemming algorithms. Technical Report MLTT-023, Xerox Research and Technology, Jan. 1996.

[154] C. Hulme, S. Maughan, and G. D. A. Brown. Memory for familiar and unfamiliar words: Evidence for a long-term memory contribution to short-term memory span. *Journal of Memory and Language*, 30:685–701, 1991.

[155] E. Hunt. The Whorfian hypothesis: A cognitive psychology perspective. *Psychological Review*, 98(3):377–389, 1991.

[156] E. Hunt, C. Lunneborg, and J. Lewis. What does it mean to be high verbal? *Cognitive Psychology*, 7:194–227, 1975.

[157] J. Hyönä. Do irregular letter combinations attract readers' attention? Evidence from fixation locations in words. *Journal of Experimental Psychology: Human Perception and Performance*, 21(1):68–81, 1995.

[158] J. Hyönä, P. Niemi, and G. Underwood. Reading long words embedded in sentences: Informativeness of word halves affects eye movements. *Journal of Experimental Psychology: Human Perception and Performance*, 15(1):142–152, 1989.

[159] K. Ishihara, T. Okada, and S. Matsui. English vocabulary recognition and production: A preliminary survey report. Technical Report ???, ???, Aug. ???

[160] A. M. Jacobs and J. Grainger. Models of visual word recognition: Sampling the state of the art. *Journal of Experimental Psychology: Human Perception and Performance*, 20(6):1311–1334, 1994.

[161] C. T. James. Vowels and consonants as targets in the search of single words. *Bulletin of the Psychonomic Society*, 2(4B):402–404, 1974.

[162] C. T. James and D. E. Smith. Sequential dependencies in letter search. *Journal of Experimental Psychology*, 85(1):56–60, 1970.

[163] L. Jin, C. Li, and S. Mehrota. Efficient similarity string joins in large data sets. Technical Report TR-DB-02-04, University of California at Irvine, Feb. 2002.

[164] J. T. Jones, B. W. Pelham, M. C. Mirenberg, and J. J. Hetts. Name letter preferences are not merely mere exposure: Implicit egotism as self-regulation. *Journal of Experimental Psychology*, 38:170–177, 2002.

[165] F. Justicia, S. Defior, S. Pelegrina, and F. J. Martos. The sources of error in Spanish writing. *Journal of Research in Reading*, 22(2):198–202, 1999.

[166] M. J. Kahana. Associative symmetry and memory theory. *Memory & Cognition*, 30(6):823–840, 2002.

[167] M. B. Karlin and G. H. Bower. Semantic category effects in visual word search. *Perception & Psychophysics*, 19(5):417–424, 1976.

[168] M. Kawakami. Effects of phonographic and phonological neighbors on katakana word recognition. In *The Second International Conference on Cognitive Science and The* 16^{th} *Annual Meeting of the Japanese Cognitive Science Society Joint Conference (ICCS/JCSS99)*. Japanese Cognitive Science Society, July 1999.

[169] M. D. Kernighan, K. W. Church, and W. A. Gale. A spelling correction program based on a noisy channel model. In *Proceedings of COLING-90*, pages 205–210, 1990.

[170] B. Kessler and R. Treiman. Syllable structure and phoneme distribution. *Journal of Memory and Language*, 37:295–311, 1997.

[171] G. C. Kinney and D. J. Showman. The relative legibility of upper case and lower case typewritten words. *Information Display*, 4:34–39, 1967.

[172] S. Kirby. Language evolution without natural selection: From vocabulary to syntax in a population of learners. Technical Report Edinburgh Occasional Papers in Linguistics, Edinburgh University, Apr. 1998.

[173] G. Kiss, C. Armstrong, R. Milroy, and J. Piper. An associative thesaurus of English and its computer analysis. In A. J. Aitken, R. W. Bailey, and N. Hamilton-Smith, editors, *The Computer and Literary Studies*, pages 153–165. Edinburgh University Press, 1973.

[174] S. Kitayama and M. Karasawa. Implicit self-esteem in Japan: Name-letters and birthday numbers. *Personality & Social Psychology Bulletin*, 23(7):736–742, 1997.

[175] K. Koda. Effects of L1 orthographic representation on L2 phonological coding strategies. *Journal of Psycholinguistic Research*, 18(2):201–220, 1989.

[176] A. Koriat. Phonetic symbolism and feeling of knowing. *Memory & Cognition*, 3(5):545–548, 1975.

[177] H. Kozima and T. Furugori. Similarity between words computed by spreading activation on an English dictionary. In *Proceedings of the* 6^{th} *Conference of the European Chapter of the Association for Computational Linguistics (EACL-93)*, pages 232–239, 1993.

[178] H. Kozima and A. Ito. Context-sensitive measurement of word distance by adaptive scaling of a semantic space. In R. Mitkov and N. Nicolov, editors, *Recent Advances in Natural Languages processing: Selected Papers from RANLP'95*, chapter 2, pages 111–124. John Benjamins Publishing Company, 1996.

[179] D. S. Kreiner and P. B. Gough. Two ideas about spelling: Rules and word-specific memory. *Journal of Memory and Language*, 29:103–118, 1990.

[180] R. Krovetz. Viewing morphology as an inference process. Technical Report UM-CS-1993-036, University of Mass-Amherst, Apr. 1993.

[181] H. Kucera and W. N. Francis. *Computational analysis of present-day American English*. Brown University Press, 1967.

[182] T. Kuennapas and A.-J. Janson. Multidimensional similarity of letters. *Perceptual and Motor Skills*, 28:3–12, 1969.

[183] K. Kukich. Spelling correction for the telecommunications network for the deaf. *Communications of the ACM*, 35(5):80–99, May 1992.

[184] K. Kukich. Techniques for automatically correcting words in text. *ACM Computing Surveys*, 24(4):377–439, 1992.

[185] P. Kundu and B. B. Chaudhuri. Error pattern in Bangla text. *International Journal of Dravidian Linguistics*, 28(2):49–88, 1999.

[186] W. Labov. The boundaries of words and their meaning. In C.-J. N. Bailey and R. W. Shuy, editors, *New ways of analyzing variation of English*, pages 340–373. Georgetown Press, 1973.

[187] K. Laitinen. *Natural naming in software development and maintenance*. PhD thesis, University of Oulu, Finland, Oct. 1995. VTT Publications 243.

[188] K. Laitinen, J. Taramaa, M. Heikkilä, and N. C. Rowe. Enhancing maintainability of source programs through disabbreviation. *Journal of Systems and Software*, 37:117–128, 1997.

[189] G. Lakoff and M. Johnson. *Metaphors We Live By*. The University of Chicago Press, 1980.

[190] B. L. Lambert, K.-Y. Chang, and P. Gupta. Effects of frequency and similarity neighborhoods on phamacists' visual perception of drug names. *Social Science and Medicine*, 57(10):1939–1955, Nov. 2003.

[191] T. K. Landauer, P. W. Foltz, and D. Laham. An introduction to latent semantic analysis. *Discourse Processes*, 25:259–284, 1998.

[192] T. K. Landauer and L. A. Streeter. Structural differences between common and rare words: Failure of equivalence assumptions for theories of word recognition. *Journal of Verbal Learning and Verbal Behavior*, 12:119–131, 1973.

[193] M. Lange and A. Content. The grapho-phonological system of written French: Statistical analysis and empirical evaluation. In *ACL'99: The 37^{th} Annual Meeting of the Association for Computational Linguistics*, pages 436–442, June 1999.

[194] D. J. Lee. Interpretation of morpheme rank ordering in L2 research. In P. S. Dale and D. Ingram, editors, *Child language: an international perspective*, chapter ???, page ??? Baltimore: University Park Press, 1981.

[195] G. Leech, P. Rayson, and A. Wilson. *Word Frequencies in Written and Spoken English*. Pearson Education, 2001.

[196] W. Li. Random texts exhibit Zipf's–law–like word frequency distribution. *IEEE Transactions on Information Theory*, 38(6):1842–1845, 1992.

[197] D. Lin. Automatic retrieval and clustering of similar words. In *Proceedings of Coling/ACL-98*, pages 768–774, 1998.

[198] A. F. Llitjós. Improving pronunciation accuracy of proper names with language origin classes. Thesis (m.s.), Carnegi Mellon University, PA, USA, Aug. 2001.

[199] R. H. Logie, S. D. Sala, V. Wynn, and A. D. Baddeley. Visual similarity effects in immediate verbal serial recall. *The Quarterly Journal of Experimental Psychology*, 53A(3):626–646, 2000.

[200] Longman. *Longman Dictionary of Contemporary English*. Longman, 2001.

[201] D. P. Lopresti and A. Tomkins. Block edit models for approximate string matching. *Theoretical Computer Science*, 181(1):159–179, 1997.

[202] E. A. Lovelace and S. S. Southall. Memory for words in prose and their locations on the page. *Memory & Cognition*, 11(5):429–434, 1983.

[203] J. A. Lucy. *Language diversity and thought: A reformulation of the linguistic relativity hypothesis*. Cambridge University Press, 1992.

[204] G. Lukatela and M. T. Turvey. Reading in two alphabets. *American Psychologist*, 53(9):1057–1072, 1998.

[205] R. Lutz and S. Greene. Measuring phonological similarity: The case of personal names. Technical Report ???, Language Analysis Systems, Inc., 2001.

[206] I. S. MacKenzie and R. W. Soukoreff. Text entry for mobile computing: Models and methods, theory and practice. *Human-Computer Interaction*, 17:147–198, 2002.

[207] M. Magnus. *What's in a Word? Studies in Phonosemantics*. PhD thesis, Norwegian University of Department of Linguistics Science and Technology, Apr. 2001.

[208] K. J. Malmberg, M. Steyvers, J. D. Stephens, and R. M. Shiffrin. Feature frequency effects in recognition memory. *Memory & Cognition*, 30(4):607–613, 2002.

[209] B. C. Malt and S. A. Sloman. Linguistic diversity and object naming by non-native speakers of English. *Bilingualism: Language and Cognition*, 6(1):47–67, 2003.

[210] B. C. Malt, S. A. Sloman, S. Gennari, M. Shi, and Y. Wang. Knowing versus naming: Similarity and the linguistic categorization of artifacts. *Journal of Memory and Language*, 40:230–262, 1999.

[211] B. C. Malt, S. A. Sloman, and S. P. Gennari. Universality and language specificity in object naming. *Journal of Memory and Language*, 49(1):20–42, 2003.

[212] W. J. Masek and M. S. Paterson. How to compute string-edit distances quickly. In D. Sankoff and J. Kruskal, editors, *Time Warps, String Edits, and Macromolecules*, chapter 14, pages 337–349. CSLI Publications, 1999.

[213] D. W. Massaro, R. L. Venezky, and G. A. Taylor. Orthographic regularity, positional frequency, and visual processing of letter strings. *Journal of Experimental Psychology: General*, 108(1):107–122, 1979.

[214] A. G. R. McClelland, R. E. Rawles, and F. E. Sinclair. The effects of search criteria and retrieval cue availability on memory for words. *Memory & Cognition*, 9(2):164–168, 1981.

[215] J. L. McClelland and D. E. Rumelhart. An interactive activation model of context effects in letter perception: Part 1. An account of basic findings. *Psychological Review*, 88(5):375–405, 1981.

[216] K. B. McDermott. The persistence of false memories in list recall. *Journal of Memory and Language*, 35:212–230, 1996.

[217] E. McKone. Short-term implicit memory for words and nonwords. *Journal of Experimental Psychology: Learning, Memory & Cognition*, 21(5):1108–1126, 1995.

[218] K. H. McWeeny, A. W. Young, D. C. Hay, and A. W. Ellis. Putting names to faces. *British Journal of Psychology*, 78:143–149, 1987.

[219] G. A. Miller, J. S. Bruner, and L. Postman. Familiarity of letter sequences and tachistoscope identification. *The Journal of General Psychology*, 50:129–139, 1954.

[220] R. Mitton. *English Spelling and the Computer*. Longman, 1996.

[221] J. Monaghan and A. W. Ellis. What exactly interacts with spelling-sound consistency in word naming? *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 28(1):183–206, 2002.

[222] A. F. Monk and C. Hulme. Errors in proofreading: Evidence for the use of word shape in word recognition. *Memory & Cognition*, 11(1):16–23, 1983.

[223] S. Monsell, K. E. Patterson, A. Graham, C. H. Hughes, and R. Milroy. Lexical and sublexical translation of spelling to sound: Strategic anticipation of lexical status. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 18(3):452–467, 1992.

[224] B. J. T. Morgan. Cluster analysis of two acoustic confusion matrices. *Perception & Psychophysics*, 13:12–24, 1973.

[225] B. J. T. Morgan, S. M. Chambers, and J. Morton. Acoustic confusion of digits in memory and recognition. *Perception & Psychophysics*, 14(2):375–383, 1973.

[226] J. Morris and G. Hirst. Lexical cohension computed by thesaural relations as an indicator of the structure of text. *Computational Linguistics*, 17(1):21–48, 1991.

[227] D. Moseley. How lack of confidence in spelling affects children's written expression. *Educational Psychology in Practice*, 5(1):42–46, 1989.

[228] S. T. Mueller, T. L. Seymour, D. E. Kieras, and D. E. Meyer. Theoretical implications of articulatory duration, phonological similarity, and phonological complexity in verbal working memory. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 29(6):1353–1380, 2003.

[229] P. Muter and E. E. Johns. Learning logographies and alphabetic codes. *Applied Cognitive Psychology*, 4:105–125, 1985.

[230] J. S. Nairne. A feature model of immediate memory. *Memory & Cognition*, 18(3):251–269, 1990.

[231] P. Nation and R. Waring. Vocabulary size, text coverage and word lists. In N. Schmitt and M. McCarthy, editors, *Vocabulary: Description, Acquisition and Pedagogy*, chapter 1.1, pages 6–19. Cambridge University Press, 1998.

[232] G. Navarro, R. Baeza-Yates, and J. ao Marcelo Azevedo Arcoverde. Matchsimile: A flexible approximate matching tool for personal names searching. *Journal of the American Society of Information Systems and Technology*, 54(1):3–15, 2003.

[233] I. Neath and J. S. Nairne. Word-length effects in immediate memory: Overwriting trace decay theory. *Psychonomic Bulletin & Review*, 2(4):429–441, 1995.

[234] D. L. Nelson and C. McEvoy. Word fragments as retrieval cues: Letter generation or search through nonsemantic memory? *American Journal of Psychology*, 97(1):17–36, 1984.

[235] D. L. Nelson, C. L. McEvoy, and T. A. Schreiber. The university of south Florida word association, rhyme and word fragment norms. Technical report, University of South Florida, Aug. 1999. `www.usf.edu/FreeAssociation`.

[236] B. K. Nirmal. *PROGRAMMING STANDARDS and GUIDELINES: COBOL edition*. Prentice-Hall, Inc, 1987.

[237] J. M. Nuttin Jr. Affective consequence of mere ownership: The name letter effect in twelve European languages. *European Journal of Social Psychology*, 17:381–402, 1987.

[238] K. Oflazer and C. Güzet. Spelling correction in agglutinative languages. Technical Report BU-CEIS-9401, Bilkent University, 1994.

[239] K. R. Paap, S. L. Newsome, and R. W. Noel. Word shape's in poor shape for the race to the lexicon. *Journal of Experimental Psychology: Human Perception and Performance*, 10(3):413–428, 1984.

[240] V. Pagel, K. Lenzo, and A. W. Black. Letter to sound rules for accented lexicon compression. In *ICSLP98*, volume 5, pages 2015–2020, 1998.

[241] J. Palmer, P. Verghese, and M. Pavel. The psychophysics of visual search. *Vision Research*, 40:1227–1268, 2000.

[242] E. Paulesu, E. McCrory, F. Fazio, L. Menoncello, N. Brunswick, S. F. Cappa, M. Cotelli, G. Cossu, F. Corte, M. Lorusso, S. Pesenti, A. Gallagher, D. Perani, C. Price, C. D. Frith, and U. Frith. A cultural effect on brain function. *Nature Neuroscience*, 3(1):91–96, Jan. 2000.

[243] R. Peereman and A. Content. Orthographic and phonological neighborhood in naming: Not all neighbors are equally influential in orthographic space. *Journal of Memory and Language*, 37:382–410, 1997.

[244] R. Peereman and A. Content. Quantitative analyses of orthography to phonology mapping in English and French. `student.vub.ac.be/~acontent/OPMapping.html`, 1998.

[245] M. Perea and E. Rosa. Does "whole word shape" play a role in visual word recognition? *Perception and Psychophysics*, 64(5):785–794, 2002.

[246] C. Perry, J. C. Ziegler, and M. Coltheart. How predictable is spelling? developing and testing metrics of phoneme-grapheme contingency. *The Quarterly Journal of Experimental Psychology*, 55A(3):897–915, 2002.

[247] J. L. Peterson. A note on undetected typing errors. *Communications of the ACM*, 29(7):633–637, 1986.

[248] R. J. Phillips. Why is lower case better? *Applied Ergonomics*, 10(4):211–214, 1979.

[249] R. J. Phillips. Searching for a target in a random arrangement of names: An eye fixation analysis. *Canadian Journal of Psychology*, 35(4):330–346, 1981.

[250] S. Pinker. *Words and Rules*. Weidenfeld & Nicolson, 1999.

[251] A. Pirkola. Morphological typology of languages in IR. *Journal of Documentation*, 57(3):330–348, 2001.

[252] M. Plauché, C. Delogu, and J. J. Ohala. Asymmetries in consonant confusion. In *Proceedings of Eurospeech 1997: 5^{th} European Conference on Speech Communication and Technology: Vol 4*, pages 2187–2190, Sept. 1997.

[253] D. C. Plaut and J. R. Booth. Individual and developmental differences in semantic priming: Empirical and computational support for a single-mechanism account of lexical processing. *Psychological Review*, 107:786–823, 2000.

[254] D. C. Plaut, J. L. McClelland, M. S. Seidenberg, and K. Patterson. Understanding normal and impaired word reading: Computational principles in quasi-regular domains. *Psychological Review*, 103(1):56–115, 1996.

[255] J. J. Pollock and A. Zamora. Collection and characterization of spelling errors in scientific and scholarly text. *Journal of the American Society for Information Science*, 34(1):51–58, 1983.

[256] M. Popovic and P. Willett. The effectiveness of stemming for natural-language access to Slovene textual data. *Journal of the American Society for Information Science*, 43(5):384–390, 1992.

[257] M. C. Potter, A. Moryadas, I. Abrams, and A. Noel. Word perception and misperception in context. *Journal of Experimental Psychology: Learning, Memory & Cognition*, 19(1):3–22, 1993.

[258] D. M. W. Powers. Applications and explanations of Zipf's law. In D. M. W. Powers, editor, *NeMLaP3/CoNLL98: New Methods in Language Processing and Computational Natural Language Learning*, pages 151–160, 1998.

[259] L. Pring. Phonological codes and functional spelling units: Reality and implications. *Perception and Psychophysics*, 30(6):573–578, 1981.

[260] K. Rastle and M. Coltheart. Whammies and double whammies: The effect of length on nonword reading. *Psychonomic Bulletin and Review*, 5:277–282, 1998.

[261] K. Rayner, K. S. Binder, J. Ashby, and A. Pollatsek. Eye movement control in reading: word predictability has little influence on initial landing position in words. *Vision Research*, 41:943–954, 2001.

[262] G. M. Reicher. Perceptual recognition as a function of meaningfulness of stimulus material. *Journal of Experimental Psychology*, 81(2):275–280, 1969.

[263] A. Rey, A. M. Jacobs, F. Schmidt-Weigand, and J. C. Ziegler. A phoneme effect in visual word recognition. *Cognition*, 68:B71–B80, 1998.

[264] A. Rey, J. C. Ziegler, and A. M. Jacobs. Graphemes are perceptual reading units. *Cognition*, 75:B1–B12, 2000.

[265] J. C. Richards. The role of vocabulary teaching. *TESOL Quarterly*, 10(1):77–89, 1976.

[266] E. Z. Rothkopf. Incidental memory for location of information in text. *Journal of Verbal Learning and Verbal Behavior*, 10:608–613, 1971.

[267] D. C. Rubin. Within word structure in the tip-of-the-tongue phenomenon. *Journal of Verbal Learning and Verbal Behavior*, 14:392–397, 1975.

[268] D. E. Rumelhart and D. A. Norman. Similating a skilled typist: A study of skilled cognitive-motor performance. *Cognitive Science*, 6:1–36, 1982.

[269] T. Sanocki. Font regularity constraints on the process of letter recognition. *Journal of Experimental Psychology: Human Perception and Performance*, 14(3):472–480, 1988.

[270] I. Schloss. Chickens and pickles. *Journal of Advertising Research*, 21(6):47–49, Dec. 1981.

[271] J. W. Schoonard and S. J. Boies. Short type: A behavior analysis of typing and text entry. *Human Factors*, 17(2):203–214, 1975.

[272] A. Scott. The vertical direction and time in Mandarin. *Australian Journal of Linguistics*, 9:295–314, 1989.

[273] E. O. Selkirk. *The Syntax of Words*. MIT Press, 1982.

[274] C. A. Sevald and G. S. Dell. The sequential cuing effect in speech production. *Cognition*, 53:91–127, 1994.

[275] L. H. Shaffer and J. Hardwick. Typing performance as a function of text. *Quarterly Journal of Experimental Psychology*, 20:360–369, 1968.

[276] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656, 1948.

[277] D. J. Shaw. A phonological interpretation of two acoustic confusion matrices. *Perception & Psychophysics*, 17(6):537–542, 1975.

[278] R. M. Shiffrin and M. Steyvers. A model for recognition memory: REM – retrieving effectively from memory. *Psychonomic Bulletin & Review*, 4(2):145–166, 1997.

[279] C. Simonyi. *Meta-Programming: A Software Production Method*. PhD thesis, Stanford University, 1977. `msdn.microsoft.com/library/techart/hunganotat.htm`.

[280] W. T. Siok and P. Fletcher. The role of phonological awareness and visual-orthographic skills in Chinese reading acquisition. *Developmental Psychology*, 37(6):886–899, 2001.

[281] J. A. Sloboda. Visual imagery and individual differences in spelling. In U. Frith, editor, *Cognitive Processes in Spelling*, chapter 11, pages 231–248. Academic Press, 1980.

[282] S. A. Sloman, M. C. Harrison, and B. C. Malt. Recent exposure affects artifact naming. *Memory & Cognition*, 30(5):687–695, 2002.

[283] R. L. Solso and P. F. Barabuto jr. Bigram and trigram frequencies and versatilities in the English language. *Behavior Research Methods & Instrumentation*, 11(5):475–484, 1979.

[284] R. L. Solso and C. L. Juel. Positional frequency and versatility of bigrams for two- through nine-letter English words. *Behavior Research Methods & Instrumentation*, 12(3):297–343, 1980.

[285] R. L. Solso and J. F. King. Frequency and versatility of letters in the English language. *Behavior Research Methods & Instrumentation*, 8(3):283–286, 1976.

[286] D. Sperber and D. Wilson. *Relevance: Communication and Cognition*. Blackwell Publishers, second edition, 1995.

[287] K. T. Spoehr and E. E. Smith. The role of orthographic and phonotactic rules in perceiving letter patterns. *Journal of Experimental Psychology: Human Perception and Performance*, 104(1):21–34, 1975.

[288] R. Sproat, A. Black, S. Chen, S. Kumar, M. Ostendorf, and C. Richards. Normalization of non-standard words: WS'99 final report. Technical Report `www.clsp.jhu.edu/ws99/projects/normal`, The Johns Hopkins University, Sept. 1999.

[289] J. Stanlaw. Two observations on culture contact and the Japanese color nomenclature system. In C. L. Hardin and L. Maffi, editors, *Color categories in thought and language*, chapter 11, pages 240–260. Cambridge University Press, 1997.

[290] K. E. Stanovich and D. W. Bauer. Experiments on the spelling-to-sound regularity effects in word recognition. *Memory & Cognition*, 6(4):410–415, 1978.

[291] C. M. Sterling. Spelling errors in context. *British Journal of Psychology*, 74:353–364, 1983.

[292] M. Steyvers. *Modeling semantic and orthographic similarity effects on memory for individual words*. PhD thesis, Indiana University, Sept. 2000.

[293] M. Steyvers and K. J. Malmberg. The effect of normative contextual variability on recognition memory. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 29(5):760–766, 2003.

[294] M. Steyvers and J. B. Tenenbaum. The large-scale structure of semantic networks: Statistical analysis and a model of semantic growth. *Cognitive Science*, submitted(???):???, Apr. 2004.

[295] G. O. Stone, M. Vanhoy, and G. C. V. Orden. Perception is a two-way street: Feedforward and feedback phonology in visual word recognition. *Journal of Memory and Language*, 36:337–359, 1997.

[296] E. Strain, K. Patterson, and M. S. Seidenberg. Semantic effects in single-word naming. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 21(5):1140–1154, 1995.

[297] L. A. Streeter, J. M. Ackroff, and G. A. Taylor. On abbreviating command names. *The Bell System Technical Journal*, 62(6):1807–1826, 1983.

[298] K. Svatko. Descriptive adjective ordering in English and Arabic. Quoted from by Celce-Murcia,[53] 1979.

[299] M. Swan and B. Smith. *Learner English*. Cambridge University Press, second edition, 2001.

[300] M. Taft and K. I. Foster. Lexical storage and retrieval of prefixed words. *Journal of Verbal Learning and Verbal Behavior*, 14:638–647, 1975.

[301] L. H. Tan and C. A. Perfetti. Phonological activation in visual identification of Chinese two-character words. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 25(2):382–393, 1999.

[302] W. J. Teahan and J. G. Cleary. The entropy of English using PPM-based models. In *IEEE Data Compression Conference*, pages 53–62. IEEE Computer Society Press, 1996.

[303] M. A. Tinker. The relative legibility of the letters, the digits, and of certain mathematical signs. *Journal of Generative Psychology*, 1:472–494, 1928.

[304] J. T. Townsend. Alphabetic confusion: A test for individuals. *Perception & Psychophysics*, 9(6):449–454, 1971.

[305] J. T. Townsend. Theoretical analysis of an alphabetic confusion matrix. *Perception & Psychophysics*, 9(1A):40–50, 1971.

[306] R. Treiman and C. Barry. Dialect and authography: Some differences between American and British spellers. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 26:1423–1430, 2000.

[307] R. Treiman and B. Kessler. Context sensitivity in the spelling of English vowels. *Journal of Memory and Language*, 47:448–468, 2002.

[308] A. Tyler and V. Evans. Reconsidering prepositional polysemy networks: The case of over. *Language*, 77(4):724–765, 2001.

[309] A. Tyler and V. Evans. *The Semantics of English Prepositions*. Cambridge University Press, 2003.

[310] T. Valentine, T. Brennen, and S. Breédart. *The Cognitive Psychology of Proper Names*. Routledge, 1996.

[311] O. Van den Bergh, S. Vrana, and P. Eelen. Letters from the heart: Affective categorization of letter combinations in typists and nontypists. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 16(6):1153–1161, 1990.

[312] F. van der Velde, A. H. C. van der Heijden, and R. Schreuder. Context-dependent migrations in visual word perception. *Journal of Experimental Psychology: Human Perception and Performance*, 15(1):133–141, 1989.

[313] W. J. B. van Heuven, T. Dijkstra, and J. Grainger. Orthographic neighborhood effects in bilingual word recognition. *Journal of Memory and Language*, 39:458–483, 1998.

[314] G. C. Van Orden. A ROWS is a ROSE: Spelling, sound, and reading. *Memory & Cognition*, 15(3):181–198, 1987.

[315] R. J. J. H. van Son and L. C. W. Pols. An acoustic model of communicative efficiency in consonants and vowels taking into account context distinctiveness. In *Proceedings of ICPhS (International Congress of Phonetic Sciences)*, page ???, ???, Aug. 2003. ???

[316] J. J. Vannest. *Morphological Effects in Visual Word Processing: Their Timecourse and Consequences for Lexical Architecture*. PhD thesis, The Ohio State University, 2001.

[317] A. G. Vartabedian. The effects of letter size, case, and generation method on CRT display search time. *Human Factors*, 13(4):363–368, 1971.

[318] R. L. Venezky. From Webster to Rice to Roosevelt: The formative years for spelling instruction and spelling reform in the U.S.A. In U. Frith, editor, *Cognitive Processes in Spelling*, chapter 1, pages 9–30. Academic Press, 1980.

[319] R. L. Venezky. *The American Way of Spelling*. Guilford Press, 1999.

[320] T. Vitale. An algorithm for high accuracy name pronunciation by parametric speech synthesizer. *Computational Linguistics*, 17(3):258–276, 1991.

[321] W. Vonk, R. Radach, and H. van Rijn. Eye guidance and the saliency of word beginnings in reading text. In A. Kennedy, R. Radach, D. Heller, and J. Pynte, editors, *Reading as a Perceptual Process*, chapter 11, pages 269–299. North-Holland, 2000.

[322] B. S. Weekes. Differential effects of number of letters on word and nonword naming latency. *Quarterly Journal of Experimental Psychology*, 50A(2):439–456, 1997.

[323] S. Wells-Jensen. *Cognitive Correlates of Linguistic Complexity: A Cross-Linguistic Comparison of Errors in Speech*. PhD thesis, State University of New York at Buffalo, U.S.A., July 1999.

[324] W. A. Wickelgren. Short-term memory for repeated and non-repeated items. *Quarterly Journal of Experimental Psychology*, 17:14–25, 1965.

[325] A. M. Wing and A. D. Baddeley. Spelling errors in handwriting: A corpus and distributional analysis. In U. Frith, editor, *Cognitive Processes in Spelling*, chapter 12, pages 251–285. Academic Press, 1980.

[326] E. Wisniewski and B. C. Love. Relations versus properties in conceptual combination. *Journal of Memory and Language*, 38:177–202, 1998.

[327] E. J. Wisniewski and D. Gentner. On the combinatorial semantics of noun pairs: Minor and major adjustments to meaning. In G. B. Simpson, editor, *Understanding word and sentence*, pages 241–284. Amsterdam: North Holland, 1991.

[328] E. J. Yannakoudakis and D. Fawthrop. The rules of spelling errors. *Information Processing & Management*, 19(2):87–99, 1983.

[329] P. N. Yianilos and K. G. Kanzelberger. The LIKEIT intelligent string comparison facility. Technical Report ???, NEC Research Institute, May 1997.

[330] E. M. Zamora, J. J. Pollock, and A. Zamora. The use of trigram analysis for spelling error detection. *Information Processing & Management*, 17(6):305–316, 1981.

[331] J. D. Zevin and M. S. Seidenberg. Age of acquisition effects in word reading and other tasks. *Journal of Memory and Language*, 47:1–29, 2002.

[332] J. Zhang and H. J. Hamilton. Learning English syllabification rules. In R. E. Mercer and E. Neufeld, editors, *Advances in Artificial Intelligence:* 12^{th} *Biennial Conference of the Canadian AI Society for the Computational Studies of Intelligence (AI'98) Proceedings*, pages 246–258, 1998.

[333] J. C. Ziegler, C. Perry, A. M. Jacobs, and M. Braun. Identical words are read differently in different languages. *Psychological Science*, 12(5):379–384, Sept. 2001.

[334] J. C. Ziegler, A. Rey, and A. M. Jacobs. Simulating individual word identification thresholds and errors in the fragmentation task. *Memory & Cognition*, 26(3):490–501, 1998.

[335] G. K. Zipf. *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology*. Addison–Wesley, 1949.