Lexical Databases for Carrier

William J. Poser
University of Pennsylvania

[This is the revised and expanded text of the paper I had prepared, but due to illness was unable to give, for the workshop “New Methods for Creating, Exploring and Disseminating Linguistic Field Data”, organized by Steven Bird under the auspices of the Talkbank Project at the University of Pennsylvania, on 6 January 2000 in Chicago, Illinois.]

1. Introduction

Research on Carrier, a dialectally diverse Athabaskan language of the central interior of British Columbia, has resulted in a fairly large amount of information stored online. The material I have includes lexical databases for seven dialects containing over 36,000 Carrier entries. Carrier is dialectally diverse, with differences not only in details of pronounciation, e.g. Stuart Lake *teł* “basket” versus Lheidli T’enneh *tił*, and in lexical items, e.g. Stuart Lake *ł ztih* “knife” versus Lheidli T’enneh *tes*, but in morphophonemics, morphology, and syntax. The dialect differences are, moreover, considered important by the communities. Although the dialects are mutually comprehensible, people consider it important to record and pass on their own dialect. There are therefore actually seven separate Carrier lexical databases, for different dialects, varying considerably in size.

The databases in which this material is kept are used both for research and to generate printed dictionaries. The database system is home-brew, developed incrementally over a decade. A similar system, though different in detail, underlies the Montana Salish (Flathead) dictionary (Thomason 1994)[1] and the dictionaries

[1] The database files are in a similar format, and similar software is used to generate TEX files from the database. These TEX files are then further processed using TEX macros and format files written by Richmond Thomason. The database itself is maintained by the linguists, Sarah Thomason and Lucy Thomason, using Epsilon, a programmable text editor similar to Emacs that runs under DOS. (Epsilon is a product of Lugaru Software. `http://www.lugaru.com`.) When a printed dictionary is to be generated, the database files are transferred to a UNIX system and the software run there.

of Oapan and Ameyaltepec Nahuatl (Amith 2002ab).[2] The purpose of this paper is to describe this system and explore its virtues and vices.

2. The Structure and Content of the Database

The database files are all pure ASCII text files. Records are delimited by extra newlines. Records are divided into fields by the field delimiter "%". Each field consists of a tag, which identifies the field, and the contents of the field, separated from the tag by a colon.[3]

A typical record is displayed in (1):

(1) A Typical Record

```
%P:tsa
%G:beaver
%IH:beaver
%C:N
%SN:Castor canadensis
%SF:animals-water
%S:JOPA/BRBI/STJA/EDFR/VESE/JEKO/MAGO
%UID:000044
%MD:2001/04/18
```

The first field, with the "P" tag for "pronounciation" is the Carrier form. The "G" field is the gloss and the "IH" field, in this case identical, is the "inverse header", used as the headword when going from English to Carrier. The "C" field contains the syntactic category, in this case "noun". The "SN" field contains the scientific name. The "SF" field contains a specification of the semantic field. This is used in generating topical indices; it is also useful when extracting words of particular types, such as kinship terms or biological terms. The "S" field identifies the sources of the data; entries are separated by slashes. In this case, the sources are all individual speakers, identified by abbreviations, but written sources may also be referred to. The last two fields are used for bookkeeping. The "UID" field contains an integer that uniquely identifies the record. This is used for integrity checks and for cross

[2] These dictionaries are generated from a single database, in SIL Shoebox format, edited using Microsoft Word under Windows. The database is copied to a GNU/Linux system for processing into a printed dictionary.

[3] This format is isomorphic to the widely used Summer Institute of Linguistics *Shoebox* format, in which the field delimiter is the backslash (\) and tags are separated from field contents by spaces. The reason for preferring the percent sign to backslash is that backslash has special meaning to numerous UNIX programs and requires quoting and other special treatment. The percent sign is much less trouble to deal with. I prefer the colon to space for two reasons. First, when fields are parsed during processing, since many fields will contain spaces, extra work will be done breaking the field into unnecessary pieces. More important than this rather minor efficiency consideration is the fact that what appears visually to be a space may in fact be a space, a tab, or another invisible character, while a colon is unambiguous.

referencing. Finally, the “MD” field contains the date on which the record was last modified.

Records for verbs tend to be more complex. In (2) we see most of the same fields as in (1), along with a number of new fields. The “VF” field indicates that this form is Imperfective Affirmative. The “ASP” field indicates that this is a customary aspect form. The “X” field gives the verb stem as *gok*. The “VA” field gives the valence prefix as *l*. The “STEMLISTED” field is used for book-keeping; it indicates that the stem of this form has been identified and that this form is listed in the entry for the stem as one of the forms containing it.

(2) A Typical Verb Record

```
%P:tanalgok
%G:he walks on all limbs back into the water
%IH:walks on all limbs back into the water, he
%VF:IA
%ASP:customary
%C:V
%X:gok
%VA:l
%S:STJA
%STEMLISTED:Y
%ES:000148
%UID:001377
%MD:1998/12/09
```

The “ES” field contains a pointer to an example sentence, which is stored in another file. Here is the entry for that example sentence:

(3) A Typical Example Sentence Record

```
%E:Dzen totsuk duni tanalgok.
%T:Every day the moose goes into the water.
%S:MAGO/STJA
%SID:000148
```

Numerous other fields are available. These include: grammatical information about verb forms, including tense/mode/negation, aspect, noun classification, stem, and valence prefix, lattitude, longitude, and map sheet (for places), pointers to images along with captions and picture credits, queries of several different types, remarks, indications of stylistic level, indications that the record contains information that is private or questionable and should therefore not be printed, grammatical notes, cultural notes, extended discussions of meanings, possessed forms of nouns, etymologies, and notes on phonetic detail. All in all, more than sixty distinct fields are in use.

So far we have only seen a need for two components to the databse: the file containing the example sentences and their translations, and the main file containing the records for individual forms like those shown above. There are two additional components to each database, a stem list and a root list, whose presence is motivated by the nature of the Carrier verb.

Carrier verbs consist of a stem, approximately the last syllable, and a set of prefixes. In general, the stem carries the main meaning of the verb. In (4) we have forms representing the tense/mode/negation paradigm of the verb “to go around in a boat”. For each form, the last syllable, which is the stem, is shown separated from the remainder, which consists of prefixes. In this case the prefixes convery information about the subject, tense/mode, and negation, except for the initial /n/, which denotes motion in a loop. We see that the stem varies considerably in form with tense, mode, and negation.

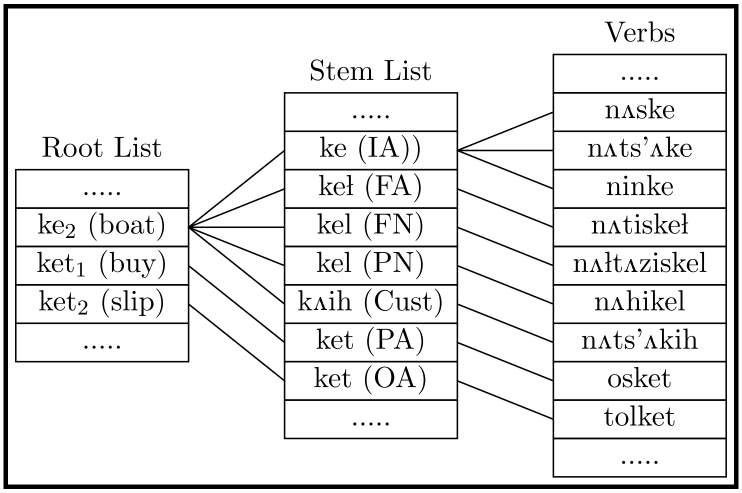
(4) Stems of “I go around in a boat” (Stuart Lake dialect)

TMA/Neg	Prefixes	Stem
Imperfective Affirmative	nʌs	ke
Imperfective Negative	nʌɬʌzʌs	koh
Perfective Affirmative	nʌsʌs	ki
Perfective Negative	nʌɬʌs	kel
Future Affirmative	nʌtis	keɬ
Future Negative	nʌɬtʌzis	kel
Optative Affirmative	nos	keʔ
Optative Negative	nʌɬʌzus	keʔ

These various stems are all associated with an abstract root, in this case *ke*, meaning “go by boat”, from which they are considered to be derived. Although there is a pattern to the changes in stems, it is complex if not irregular, and so someone learning the language must to a considerable extent simply memorize the stem set for each verb.

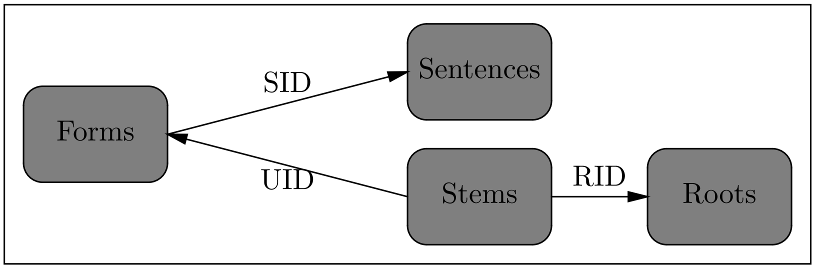
The database therefore contains, in addition to the individual verb forms, a table of verb stems and a table of verb roots. Each verb stem record contains pointers to all of the actual verb forms in which it has been found as well as to the root with which it is associated. At the level of individual forms, we may represent this relationship as in (5). For example, the root ke_2 “go by boat” has an Imperfective Affirmative stem /ke/ which is attested in three verb forms.

(5) Relationships of Verbs in a Carrier Dictionary



The same relationship is ilustrated in (6) now at the level of the components of the database. This diagram shows that the collection of forms contains pointers, in the form of S(entenc)e IDs, to example sentences. The stem list contains pointers, in the form of U(nique) IDs, to individual verb forms, and it also contains pointers, in the form of R(oot) IDs, to verb roots.

(6) The Relational Structure of a Carrier Database

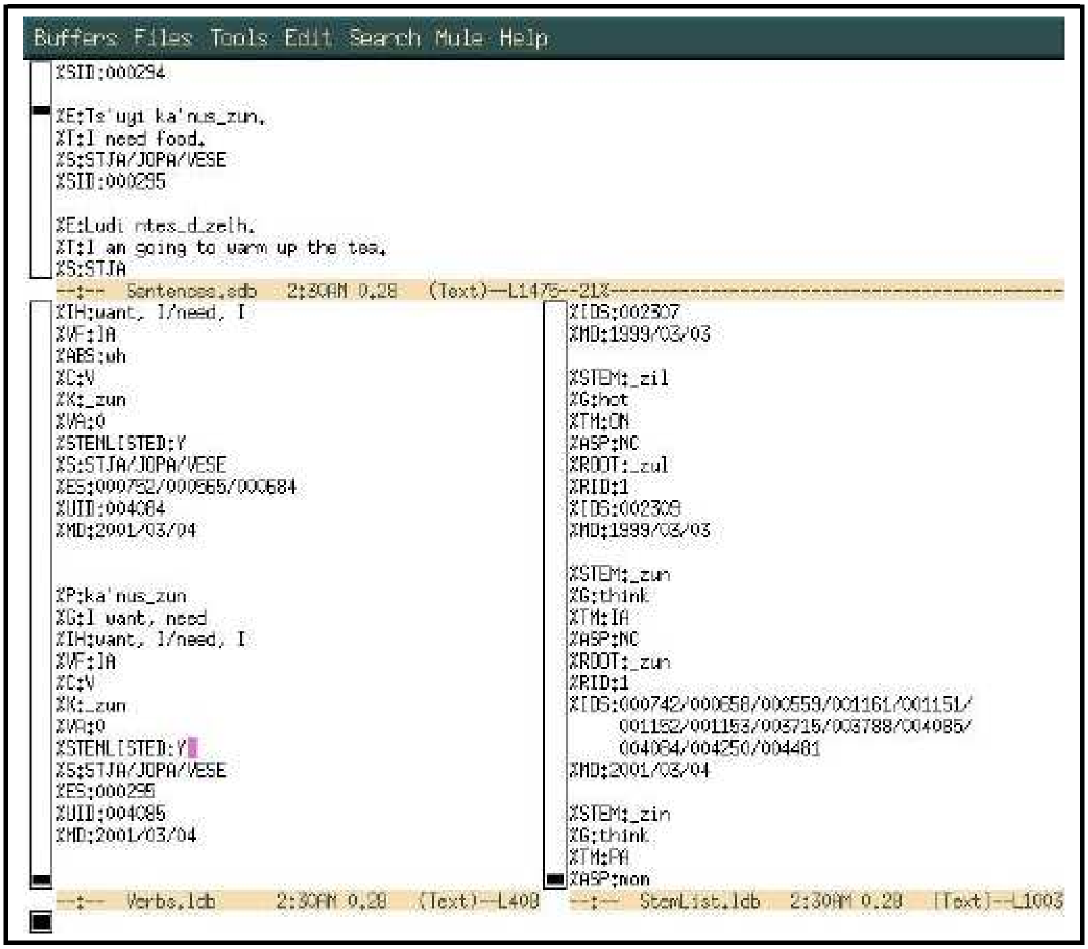


3. Modifying the Database

Since the database consists of ordinary text files, it can be modified by any text editor. In practice, this is almost always Emacs (Stallman 1984), an interpreter for a dialect of Lisp with special primitives for text editing. Since Emacs really is a programming language interpreter, it is fully programmable. It is also possible to bind any function to a combination of keystrokes. Functions written in Emacs Lisp facilitate working with the database by providing such capabilities as insertion of templates for new records of particular types with certain information already filled in, a variety of specialized searches, and shortcuts for modification of selected properties of the current record. For example, typing “Control-x t” updates the time stamp in the modification time field of the current record.

Another useful facility of Emacs is its ability recursively to split the screen both vertically and horizontally, allowing for the simultaneous display of multiple windows. In (7) we have a screen shot of Emacs in use editing a database. There are three Emacs-internal windows open. The lower left window displays a portion of the verb database.[4] The lower right window displays the portion of the stem database containing the entry for the stem of the verb in the lower left window. The top window is open to the sentence database and displays an example sentence associated with the verb in the lower left window. This configuration would be useful in adding verbs to the database. If the new verb form comes with an example sentence, the example sentence would simultaneously be added to the sentence database and the sentence ID entered in the verb form's record. If the stem of the verb form can be identified, the verb form's ID can be added to the record for the stem, if it is already in the database. If not, the new stem can be added.

(7) EMACS with Three Windows Open



```
%SID:000294

%E:Ts'ugi ka'nus_zun.
%T:I need food.
%S:STJA/JOPA/VESE
%SID:000295

%E:Ludi ntes_d_zelh.
%T:I am going to warm up the tea.
%S:STJA
--:--  Sentences.sdb   2:30AM 0.28   (Text)--L1475--21%------------------------------------------
```

```
%IH:want, I/need, I
%VF:IA
%ABS:uh
%C:V
%K:_zun
%VA:0
%STEMLISTED:Y
%S:STJA/JOPA/VESE
%ES:000752/000565/000684
%UID:004084
%MD:2001/03/04


%P:ka'nus_zun
%G:I want, need
%IH:want, I/need, I
%VF:IA
%C:V
%K:_zun
%VA:0
%STEMLISTED:Y
%S:STJA/JOPA/VESE
%ES:000295
%UID:004085
%MD:2001/03/04

--:--  Verbs.ldb      2:30AM 0.28   (Text)--L408
```

```
%IDS:002307
%MD:1999/03/03

%STEM:_zil
%G:hot
%TM:ON
%ASP:NC
%ROOT:_zul
%RID:1
%IDS:002308
%MD:1999/03/03

%STEM:_zun
%G:think
%TM:IA
%ASP:NC
%ROOT:_zun
%RID:1
%IDS:000742/000658/000559/001161/001151/
     001152/001153/003715/003788/004085/
     004084/004250/004481
%MD:2001/03/04

%STEM:_zin
%G:think
%TM:PA
%ASP:non
--:--  StemList.ldb   2:30AM 0.28   [Text]--L1003
```

[4] In some cases it has proved to be convenient to keep verbs and non-verbs in separate files. For simplicity's sake, this is not reflected in our discussion of the organization of the database.

Emacs is normally not invoked directly. Instead, a database is edited by executing a shell script, which in turn invokes Emacs. The shell script backs up the database before invoking Emacs and computes certain information used by Emacs. At the end of a session, when Emacs returns, the shell script performs some integrity checks, makes a record of the differences between the previous version of the database and the current one, and performs some clean up.

4. Printed Dictionaries

A variety of printed dictionaries are generated from the databases. From time to time specialized dictionaries, such as lists of biological terms, have been printed, but most of the time what is desired is a comprehensive dictionary. A few sections are fixed, that is, consist of information not generated automatically from the database:
(8) Fixed Sections of Printed Dictionaries

- Explanation of the Writing System
- Grammatical Information
- Alphabetical Order
- Abbreviations for Grammatical Categories
- Appendices (Numbers, Months, Days)
- References

The appendices on numbers, months, and days are provided because language teachers and parents of small children like them.[5] The inclusion of the appendix on numbers also provides the opportunity to explain the rules for generation of numbers from their basic components as well as the classificatory system, according to which numbers and some quantifiers take different forms depending on which of five categories the thing counted belongs to.

The bulk of the material, however, is derived from the database.

[5] A common belief of both parents and language teachers is that the language material appropriate for young children is the same as what they concentrate on in English in preschool and kindergarten, namely numbers, dates, and colors.

(9) Sections of Printed Dictionaries Generated from Database

- Carrier to English
- English to Carrier
- Topical Index
- Root List (By English Gloss)
- Root List (Alphabetical)
- Stem List
- Stems Sorted By Root
- Notes on Stems
- Affix List
- Index of Affixes by Gloss
- Index of Scientific Names
- Index of Place Names
- Index of English Place Names
- Index of Loanwords
- Credits for Illustrations

In addition to the usual main sections, there are also quite a few indices, which group together words of certain types and simplify finding them. The index of scientific names, for example, is useful to biologists.

Typical pages from the Carrier-to-English and English-to-Carrier sections of a dictionary are illustrated in (10) and (11). A topical index page is illustrated in (12). The various indices are exemplified by (13), which contains a page from an index of scientific names. There are also lists of roots and lists of stems, organized alphabetically and by root. These provide a means of looking up a verb form that is not in the dictionary. If the user is sufficiently knowledgable, he may be able to determine what stem of what root he is dealing with and on this basis analyze the form. Typical pages from the root and stem lists are illustrated in (14)-(17).

-chaike **chunlai**

-chaike N Plural of **-chai**, q.v..
chaimun N Chinese person, Oriental person. Etymology: Loan from English **Chinaman**.
chaimun ye'ulht'es-i N wok. Etymology: Literally, "the thing that a Chinese person uses to fry with".
chainamun N Chinese person. Etymology: Loan from English **chinaman**.
chainya V [0-ya < ya$_1$] you (1) were conceived. [PA]
-chak N ribs.
-chakesgwulh N navel.
-chak'ests'oh N armpit.
chalhts'ul N baby. Duoplural: chalhts'ulne
chalhyal N basket. A large basket used for storage or in which berries are accumulated. Normally carried on the back.

A **tilh** and a **chalhyal**.

chan N rain. **(1)** Chan tsul 'uja. 'It stopped raining.'
chanjo N newly sexually mature cow moose.
chantoo N rainwater.
-chanulhyeh N tumor in belly, inflamed appendix.
-che N sleep. **(1)** Duche tezluz. 'He wet his bed.' **(2)** Duche iluz. 'He wet his bed.' **(3)** Duche yalhtuk. 'He talks in his sleep.'
-chela N fish tail.
chilkot'in N Chilcotin person.
chilh N young man. Duoplurals: chilhuke, chilhne, chilhke
chilhuke N Irregular plural of **chilh**, q.v..
chintoh N forest, bush.
-chistl'oh N younger brother. Duoplural: -chistl'ohke
-cho **SUFFIX** big. **(1)** Lhukwcho. 'Big fish.' **(2)** Nawhulnukcho. 'Bishop [big priest].' **(3)** Bets'ulh'uzcho. 'Sledgehammer [big hammer].'
chos N canoe paddle.
-chul N younger brother.
-chun N handle. **(1)** Tsetselh buchun. 'The axe's handle.'
-chun N trunk of tree.
chundoo N Lodgepole Pine, locally known as "Jack Pine", which standardly refers to **Pinus banksiana**. *[Pinus contorta latifolia]*

chundoo — Lodgepole Pine

chundulkw'uz N woodpecker.
chunih N marten. *[Martes americana]*
chunihcho N fisher. *[Martes pennanti]* Etymology: "big marten".
chunlai N salamander, lizard. The only species of salamander found in the region

– 30 –

(10)

basket **belongs to you (1), it**

paratively narrow and deep. When collecting berries, a **tilh** will normally be emptied into a **chalhyal**.
basket. N chalhyal. A large basket used for storage or in which berries are accumulated. Normally carried on the back.
basket, soapberry. N nawusts'ai. Used for collecting **nawus** soapberries.
basket, tray-like. N tl'usts'ai. In contrast to a **tilh**, which is narrow and deep, a **tl'usts'ai** is wide and shallow. When collecting berries, it is typically emptied into a **chalhyal**.
bat. N 'ut'az.
bat. *[Myotis lucifugus]* N liyabdut'ai. The only kind of bat found in the region is the Little Brown Myotis.
bath, I take a. V toonasya. [IA] [0-ya < ya$_1$]
bath, one takes a. V toonats'uyah. [IA-customary] [0-yah < ya$_1$]
bath, they (2) are taking a. V toonahu'as. [IA] [0-'as < 'as$_1$]
bath, we (3+) take a bath. V toonats'udelh. [IA] [0-delh < del$_1$]
bathroom. N tsanbayoh.
bathroom, I am going to the. V 'un natesda. [FA] [0-da] *[polite euphemism]*
beads. N kw'usul.
beans. N lubinus.
bearpaw snowshoes. N suske.
bearpaw snowshoes. N shaske.
beat him soundly with a stick, he. V yayalht'o. [class: lro-u] [PA] [lh-t'o < t'o$_3$]
beautiful, it is. V dinzoo. [abs: **d**] [IA] [0-zoo < zoo$_1$]
beaver. *[Castor canadensis]* N tsa.
beaver, baby. N tsayaz.
beaver, big. N tsati.
beaver, female. N tsa'at.
beaver, old. N tsati.
beaver, young. N tsatsul.
beaver dam. N 'ulh.
beaver dam. N lht'at.
beaver lodge. N tsaken.
beaver path under the ice. N tunyohtsati.
beaver paws. N tsake.
beaver tail. N tsache.
became, he. V suli. [PA] [0-li]
became, I. V susdli. [PA] [0-li < li$_2$]
becomes, it. V wuleh. [OA] [0-leh < li$_2$]
bed. N lili.
bed of spruce boughs. N tel.
bed sheets. N lilik'usdla.
bedpan. N benats'udutsun.
bedroom. N ts'uztezbayoh.
bedspread. N lilik'usulhchooz.
bee. *[Apis mellifera]* N ts'ihna.
beehive. N ts'ihnat'o.
beehive burner. N beduzk'un.
beer. N hawus.
beer. N too nanteldzoos.
beets. N lhits'e.
before. **COMP** whutso. This is the postposition **tso** "before" inflected for areal object.
behind. **PP** -k'oh.
behind. **PP** t'ak.
behind the house. **ADV** koont'ak.
bell. N luglos.
belly. N -but.
bell-ringer. N luglosman. The man who rings the church bell.
belongs to him, it. V be'ilhdzun. [IA] [lh-dzun < dzun$_1$]
belongs to it, it. V whe'ilhdzun. [abs: **wh**] [IA] [lh-dzun < dzun$_1$]
belongs to me, it. V se'ilhdzun. [IA] [lh-dzun < dzun$_1$]
belongs to them, it. V hube'ilhdzun. [IA] [lh-dzun < dzun$_1$]
belongs to us, it. V nehoolhdzun. [abs: **wh**] [IA] [lh-dzun < dzun$_1$]
belongs to us (2+), it. V ne'ilhdzun. [IA] [lh-dzun < dzun$_1$]
belongs to you (1), it. V nye'ilhdzun. [IA] [lh-dzun]

– 146 –

(11)

Topical Index

beaver path under the ice tunyohtsati
black bear den sus'an
dam, beaver 'ulh
dam, beaver lht'at
leech hoolht'ukw
leech hoot'ub
moose tracks dunik'oh
tracks, grizzly bear shask'oh
tracks, rabbit gohk'oh

2. Fish and Shellfish

2.1. General

fish lhukw
fish, big lhukwcho

2.2. Fish

burbot tsintel
char bet
Dolly Varden tsabai
kokanee gesul
lake trout bet
ling cod tsintel
Mountain Whitefish lhoo
Red Salmon talukw
Salmon, Sockeye talukw
Sockeye Salmon talukw
Squawfish, Northern whusi
Sturgeon, White lhkw'encho
sucker, Largescale goosibai
trout, Rainbow duk'ai
whitefish lhewh

2.3. Parts

backbone of fish lhukwyun
bones of fish lhukwghak
fin lhukwghak
scales of fish -goos
skeleton of fish lhukwghak
tail, fish -chela

2.4. Miscellaneous

slime, fish lhokwtl'us

3. Amphibians And Reptiles

Frog, Spotted dulkw'ah
lizard chunlai
Spotted Frog dulkw'ah
salamander chunlai
snake tl'ughus
Toad, Western tsasdli
Western Toad tsasdli

4. Birds

4.1. General

American Crow datsan
Bald Eagle tsebalyan
Bluejay (Steller's Jay) tehgwuzeh
bird dut'ai
bird, little dut'aiyaz
bird, little gagiyaz
crane delh
crow datsan
duck dut'ai
Fool Hen nat'oh
Grebe, Red-Necked ts'olh
Grouse, Spruce nat'oh
goose khoh
Helldiver ts'olh
hummingbird ts'unalhduz
Jay, Canadian gwuzeh
Jay, Gray gwuzeh
jay, Steller's tehgwuzeh
loon dadzi
Mallard Duck t'ugicho
owl musdzoon
Red-Necked Grebe ts'olh
Robin, American sewh
Ruffed grouse 'utsut
Spruce Grouse nat'oh
Steller's Jay tehgwuzeh
sandpiper wedlew
seagull besk'i
swan ts'incho
Tree Swallow 'uschas
Whiskey Jack gwuzeh
woodpecker chundulkw'uz

– 245 –

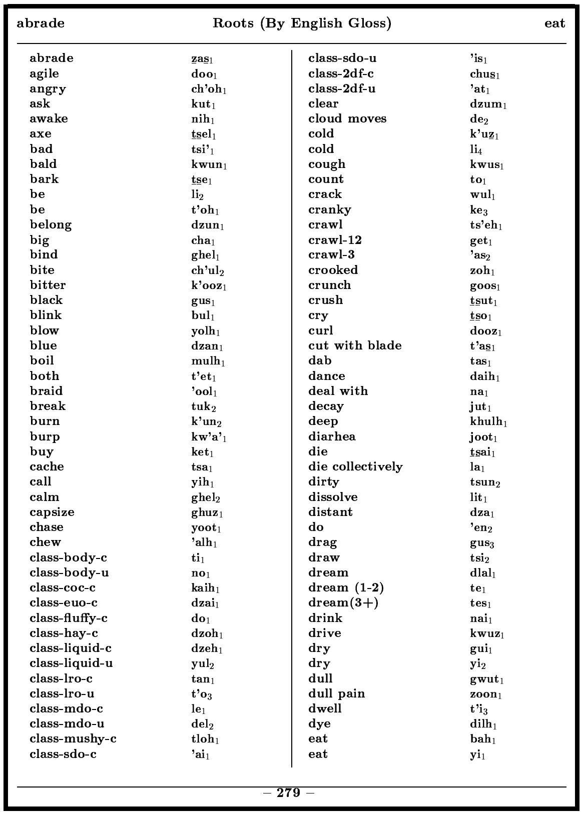
(12)

Acer glabrum **Index of Scientific Names** **Vaccinium myrtilloides?**

Acer glabrum	**khasdzoon**
Acipenser transmontanus	**lhkw'encho**
Alces alces andersoni	**duni**
Alnus viridis	**k'us**
Ambystoma macrodactylum	**chunlai**
Amelanchier alnifolia	**k'enchun**
Anas platyrhynchos	**t'ugicho**
Apis mellifera	**ts'ihna**
Betula glandulosa glandulifera	**whuzaz**
Betula occidentalis	**nach'ulh**
Betula papyrifera	**k'ui**
Bryoria lanestris	**dohgha**
Bufo boreas	**tsasdli**
Canis latrans	**tintulhi**
Canis lupus	**yus**
Castilleja miniata	**ditnikkwun**
Castor canadensis	**tsa**
Catostomus macrocheilus	**goosibai**
Cicuta douglasii	**honganicho**
Cornus stolonifera	**k'entsik**
Corvus brachyrhynchos	**datsan**
Cyanocitta stelleri	**tehgwuzeh**
Dendragapus canadensis	**nat'oh**
Erethizon dorsatum	**ts'it**
Felis concolor	**booscho**
Felis domesticus	**boos**
Felis lynx	**wasi**
Fragaria species	**'indzi**
Gavia immer	**dadzi**
Glaucomys sabrinus alpinus	**ts'unalhbuz**
Gulo gulo	**noostel**
Haliaeetus leucocephalus	**tsebalyan**
Heracleum lanatum	**goos**
Juniperus communis	**datsan'algut**
Larix laricina	**ts'oobiz**
Ledum groenlandicum	**'uyak'unulh'a**
Ledum groenlandicum	**yak'unulh'a**
Lota lota	**tsintel**
Lutra canadensists'olh	**tsis**
Marmota caligata	**dutni**
Marmota monax	**k'ani**
Martes americana	**chunih**
Martes pennanti	**chunihcho**
Mustela vison	**telhjoos**
Myotis lucifugus	**liyabdut'ai**
Oncorhynchus mykiss	**duk'ai**
Oncorhynchus nerka	**gesul**
Oncorhynchus nerka	**talukw**
Ondatra zibethicus	**tsek'et**
Perisoreus canadensis	**gwuzeh**
Pinus contorta latifolia	**chundoo**
Podiceps grisegena	**ts'olh**
Populus tremuloides	**tl'ughus**
Populus tremuloides	**tl'ughusyaz**
Populus trichocarpa	**landooz**
Porphyra abbottae Krishnamurthy	**lhaga'as**
Prunus emerginata	**nulgoosmai**
Pseudotsuga menziesii glauca	**chuntsi**
Ptychocheilus oregonensis	**whusi**
Rana pretiosa	**dulkw'ah**
Rangifer tarandus	**whudzih**
Rosa nutkana	**whuschun**
Salix species	**k'uidlih**
Salvelinus malma	**tsabai**
Salvelinus namaycush	**bet**
Shepherdia canadensis	**nawus**
Shepherdia canadensis	**nuwus**
Sorbus sitchensis	**ch'ekw**
Sphagnum capillaceum	**ts'al**
Tachycineta bicolor	**'uschas**
Tamiasciurus hudsonicus colum.	**nats'ildelh**
Thamnophis sirtalis	**tl'ughus**
Turdus migratorius	**sewh**
Ursus americanus	**sus**
Vaccinium myrtilloides?	**'ilhtsul**

– 320 –

(13)

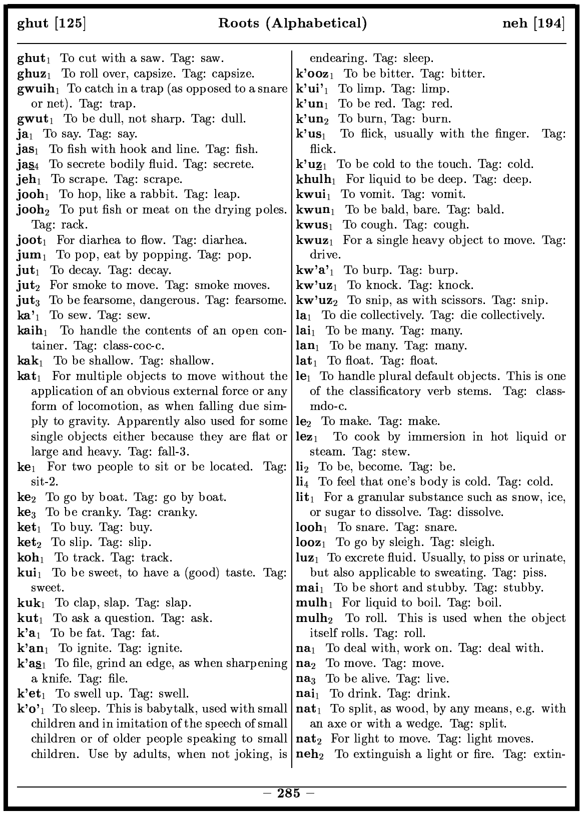


abrade **Roots (By English Gloss)** eat

abrade	**z̲as̲$_1$**
agile	**doo$_1$**
angry	**ch'oh$_1$**
ask	**kut$_1$**
awake	**nih$_1$**
axe	**t̲s̲el$_1$**
bad	**tsi'$_1$**
bald	**kwun$_1$**
bark	**t̲s̲e$_1$**
be	**li$_2$**
be	**t'oh$_1$**
belong	**dzun$_1$**
big	**cha$_1$**
bind	**ghel$_1$**
bite	**ch'ul$_2$**
bitter	**k'ooz$_1$**
black	**gus$_1$**
blink	**bul$_1$**
blow	**yolh$_1$**
blue	**dzan$_1$**
boil	**mulh$_1$**
both	**t'et$_1$**
braid	**'ool$_1$**
break	**tuk$_2$**
burn	**k'un$_2$**
burp	**kw'a'$_1$**
buy	**ket$_1$**
cache	**tsa$_1$**
call	**yih$_1$**
calm	**ghel$_2$**
capsize	**ghuz$_1$**
chase	**yoot$_1$**
chew	**'alh$_1$**
class-body-c	**ti$_1$**
class-body-u	**no$_1$**
class-coc-c	**kaih$_1$**
class-euo-c	**dzai$_1$**
class-fluffy-c	**do$_1$**
class-hay-c	**dzoh$_1$**
class-liquid-c	**dzeh$_1$**
class-liquid-u	**yul$_2$**
class-lro-c	**tan$_1$**
class-lro-u	**t'o$_3$**
class-mdo-c	**le$_1$**
class-mdo-u	**del$_2$**
class-mushy-c	**tloh$_1$**
class-sdo-c	**'ai$_1$**
class-sdo-u	**'is$_1$**
class-2df-c	**chus̲$_1$**
class-2df-u	**'at$_1$**
clear	**dzum$_1$**
cloud moves	**de$_2$**
cold	**k'uz̲$_1$**
cold	**li$_4$**
cough	**kwus$_1$**
count	**to$_1$**
crack	**wul$_1$**
cranky	**ke$_3$**
crawl	**ts'eh$_1$**
crawl-12	**get$_1$**
crawl-3	**'as$_2$**
crooked	**zoh$_1$**
crunch	**goos$_1$**
crush	**t̲s̲ut$_1$**
cry	**t̲s̲o$_1$**
curl	**dooz$_1$**
cut with blade	**t'as̲$_1$**
dab	**tas$_1$**
dance	**daih$_1$**
deal with	**na$_1$**
decay	**jut$_1$**
deep	**khulh$_1$**
diarhea	**joot$_1$**
die	**t̲s̲ai$_1$**
die collectively	**la$_1$**
dirty	**tsun$_2$**
dissolve	**lit$_1$**
distant	**dza$_1$**
do	**'en$_2$**
drag	**gus$_3$**
draw	**tsi$_2$**
dream	**dlal$_1$**
dream (1-2)	**te$_1$**
dream(3+)	**tes$_1$**
drink	**nai$_1$**
drive	**kwuz$_1$**
dry	**gui$_1$**
dry	**yi$_2$**
dull	**gwut$_1$**
dull pain	**zoon$_1$**
dwell	**t'i$_3$**
dye	**dilh$_1$**
eat	**bah$_1$**
eat	**yi$_1$**

– **279** –

(14)

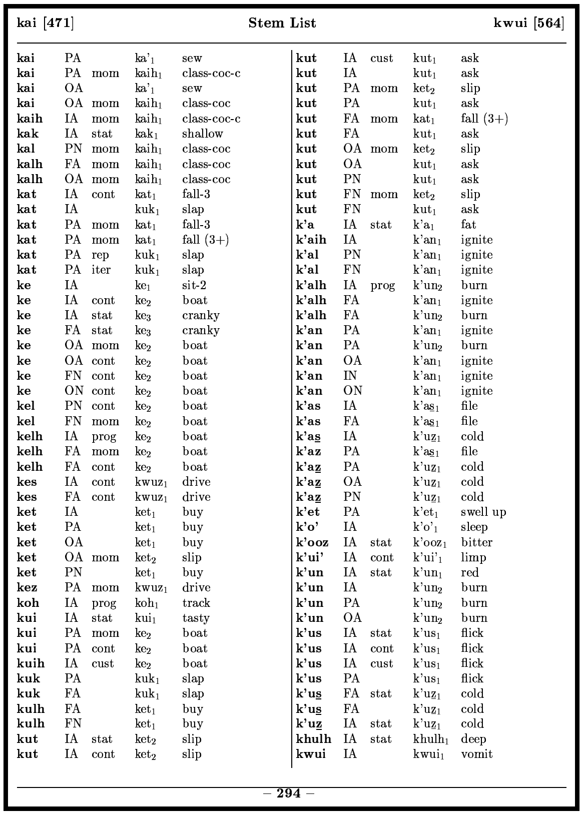


ghut [125] **Roots (Alphabetical)** neh [194]

ghut$_1$ To cut with a saw. Tag: saw.
ghuz$_1$ To roll over, capsize. Tag: capsize.
gwuih$_1$ To catch in a trap (as opposed to a snare or net). Tag: trap.
gwut$_1$ To be dull, not sharp. Tag: dull.
ja$_1$ To say. Tag: say.
jas$_1$ To fish with hook and line. Tag: fish.
jas̲$_4$ To secrete bodily fluid. Tag: secrete.
jeh$_1$ To scrape. Tag: scrape.
jooh$_1$ To hop, like a rabbit. Tag: leap.
jooh$_2$ To put fish or meat on the drying poles. Tag: rack.
joot$_1$ For diarhea to flow. Tag: diarhea.
jum$_1$ To pop, eat by popping. Tag: pop.
jut$_1$ To decay. Tag: decay.
jut$_2$ For smoke to move. Tag: smoke moves.
jut$_3$ To be fearsome, dangerous. Tag: fearsome.
ka'$_1$ To sew. Tag: sew.
kaih$_1$ To handle the contents of an open container. Tag: class-coc-c.
kak$_1$ To be shallow. Tag: shallow.
kat$_1$ For multiple objects to move without the application of an obvious external force or any form of locomotion, as when falling due simply to gravity. Apparently also used for some single objects either because they are flat or large and heavy. Tag: fall-3.
ke$_1$ For two people to sit or be located. Tag: sit-2.
ke$_2$ To go by boat. Tag: go by boat.
ke$_3$ To be cranky. Tag: cranky.
ket$_1$ To buy. Tag: buy.
ket$_2$ To slip. Tag: slip.
koh$_1$ To track. Tag: track.
kui$_1$ To be sweet, to have a (good) taste. Tag: sweet.
kuk$_1$ To clap, slap. Tag: slap.
kut$_1$ To ask a question. Tag: ask.
k'a$_1$ To be fat. Tag: fat.
k'an$_1$ To ignite. Tag: ignite.
k'as̲$_1$ To file, grind an edge, as when sharpening a knife. Tag: file.
k'et$_1$ To swell up. Tag: swell.
k'o'$_1$ To sleep. This is babytalk, used with small children and in imitation of the speech of small children or of older people speaking to small children. Use by adults, when not joking, is endearing. Tag: sleep.
k'ooz$_1$ To be bitter. Tag: bitter.
k'ui'$_1$ To limp. Tag: limp.
k'un$_1$ To be red. Tag: red.
k'un$_2$ To burn, Tag: burn.
k'us$_1$ To flick, usually with the finger. Tag: flick.
k'uz̲$_1$ To be cold to the touch. Tag: cold.
khulh$_1$ For liquid to be deep. Tag: deep.
kwui$_1$ To vomit. Tag: vomit.
kwun$_1$ To be bald, bare. Tag: bald.
kwus$_1$ To cough. Tag: cough.
kwuz$_1$ For a single heavy object to move. Tag: drive.
kw'a'$_1$ To burp. Tag: burp.
kw'uz$_1$ To knock. Tag: knock.
kw'uz$_2$ To snip, as with scissors. Tag: snip.
la$_1$ To die collectively. Tag: die collectively.
lai$_1$ To be many. Tag: many.
lan$_1$ To be many. Tag: many.
lat$_1$ To float. Tag: float.
le$_1$ To handle plural default objects. This is one of the classificatory verb stems. Tag: class-mdo-c.
le$_2$ To make. Tag: make.
lez$_1$ To cook by immersion in hot liquid or steam. Tag: stew.
li$_2$ To be, become. Tag: be.
li$_4$ To feel that one's body is cold. Tag: cold.
lit$_1$ For a granular substance such as snow, ice, or sugar to dissolve. Tag: dissolve.
looh$_1$ To snare. Tag: snare.
looz$_1$ To go by sleigh. Tag: sleigh.
luz$_1$ To excrete fluid. Usually, to piss or urinate, but also applicable to sweating. Tag: piss.
mai$_1$ To be short and stubby. Tag: stubby.
mulh$_1$ For liquid to boil. Tag: boil.
mulh$_2$ To roll. This is used when the object itself rolls. Tag: roll.
na$_1$ To deal with, work on. Tag: deal with.
na$_2$ To move. Tag: move.
na$_3$ To be alive. Tag: live.
nai$_1$ To drink. Tag: drink.
nat$_1$ To split, as wood, by any means, e.g. with an axe or with a wedge. Tag: split.
nat$_2$ For light to move. Tag: light moves.
neh$_2$ To extinguish a light or fire. Tag: extin-

– **285** –

(15)

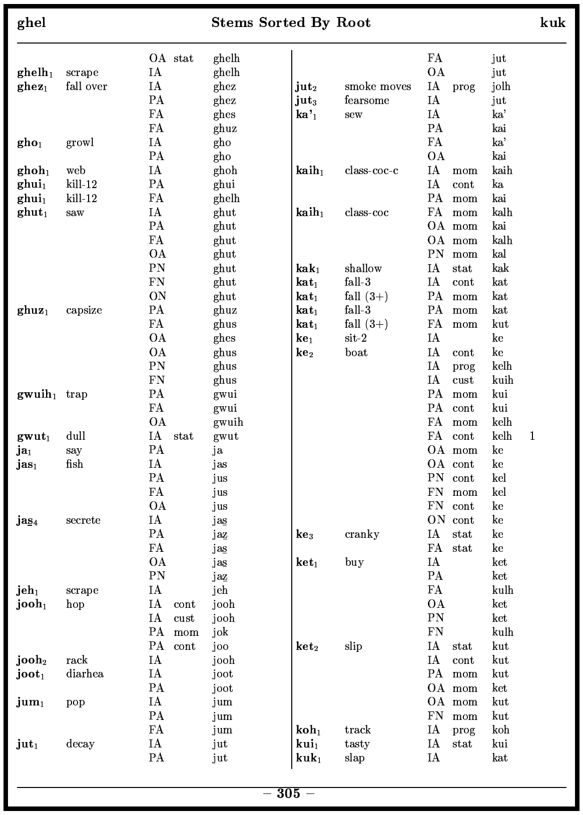


kai [471] **Stem List** kwui [564]

kai	PA		ka'$_1$	sew
kai	PA	mom	kaih$_1$	class-coc-c
kai	OA		ka'$_1$	sew
kai	OA	mom	kaih$_1$	class-coc
kaih	IA	mom	kaih$_1$	class-coc-c
kak	IA	stat	kak$_1$	shallow
kal	PN	mom	kaih$_1$	class-coc
kalh	FA	mom	kaih$_1$	class-coc
kalh	OA	mom	kaih$_1$	class-coc
kat	IA	cont	kat$_1$	fall-3
kat	IA		kuk$_1$	slap
kat	PA	mom	kat$_1$	fall-3
kat	PA	mom	kat$_1$	fall (3+)
kat	PA	rep	kuk$_1$	slap
kat	PA	iter	kuk$_1$	slap
ke	IA		ke$_1$	sit-2
ke	IA	cont	ke$_2$	boat
ke	IA	stat	ke$_3$	cranky
ke	FA	stat	ke$_3$	cranky
ke	OA	mom	ke$_2$	boat
ke	OA	cont	ke$_2$	boat
ke	FN	cont	ke$_2$	boat
ke	ON	cont	ke$_2$	boat
kel	PN	cont	ke$_2$	boat
kel	FN	mom	ke$_2$	boat
kelh	IA	prog	ke$_2$	boat
kelh	FA	mom	ke$_2$	boat
kelh	FA	cont	ke$_2$	boat
kes	IA	cont	kwuz$_1$	drive
kes	FA	cont	kwuz$_1$	drive
ket	IA		ket$_1$	buy
ket	PA		ket$_1$	buy
ket	OA		ket$_1$	buy
ket	OA	mom	ket$_2$	slip
ket	PN		ket$_1$	buy
kez	PA	mom	kwuz$_1$	drive
koh	IA	prog	koh$_1$	track
kui	IA	stat	kui$_1$	tasty
kui	PA	mom	ke$_2$	boat
kui	PA	cont	ke$_2$	boat
kuih	IA	cust	ke$_2$	boat
kuk	PA		kuk$_1$	slap
kuk	FA		kuk$_1$	slap
kulh	FA		ket$_1$	buy
kulh	FN		ket$_1$	buy
kut	IA	stat	ket$_2$	slip
kut	IA	cont	ket$_2$	slip
kut	IA	cust	kut$_1$	ask
kut	IA		kut$_1$	ask
kut	PA	mom	ket$_2$	slip
kut	PA		kut$_1$	ask
kut	FA	mom	kat$_1$	fall (3+)
kut	FA		kut$_1$	ask
kut	OA	mom	ket$_2$	slip
kut	OA		kut$_1$	ask
kut	PN		kut$_1$	ask
kut	FN	mom	ket$_2$	slip
kut	FN		kut$_1$	ask
k'a	IA	stat	k'a$_1$	fat
k'aih	IA		k'an$_1$	ignite
k'al	PN		k'an$_1$	ignite
k'al	FN		k'an$_1$	ignite
k'alh	IA	prog	k'un$_2$	burn
k'alh	FA		k'an$_1$	ignite
k'alh	FA		k'un$_2$	burn
k'an	PA		k'an$_1$	ignite
k'an	PA		k'un$_2$	burn
k'an	OA		k'an$_1$	ignite
k'an	IN		k'an$_1$	ignite
k'an	ON		k'an$_1$	ignite
k'as	IA		k'as̲$_1$	file
k'as	FA		k'as̲$_1$	file
k'as̲	IA		k'uz̲$_1$	cold
k'az	PA		k'as̲$_1$	file
k'az̲	PA		k'uz̲$_1$	cold
k'az̲	OA		k'uz̲$_1$	cold
k'az̲	PN		k'uz̲$_1$	cold
k'et	PA		k'et$_1$	swell up
k'o'	IA		k'o'$_1$	sleep
k'ooz	IA	stat	k'ooz$_1$	bitter
k'ui'	IA	cont	k'ui'$_1$	limp
k'un	IA	stat	k'un$_1$	red
k'un	IA		k'un$_2$	burn
k'un	PA		k'un$_2$	burn
k'un	OA		k'un$_2$	burn
k'us	IA	stat	k'us$_1$	flick
k'us	IA	cont	k'us$_1$	flick
k'us	IA	cust	k'us$_1$	flick
k'us	PA		k'us$_1$	flick
k'us̲	FA	stat	k'uz̲$_1$	cold
k'us̲	FA		k'uz̲$_1$	cold
k'uz̲	IA	stat	k'uz̲$_1$	cold
khulh	IA	stat	khulh$_1$	deep
kwui	IA		kwui$_1$	vomit

– **294** –

(16)



ghel **Stems Sorted By Root** kuk

		OA	stat	ghelh	
ghelh$_1$	scrape	IA		ghelh	
ghez$_1$	fall over	IA		ghez	
		PA		ghez	
		FA		ghes	
		FA		ghuz	
gho$_1$	growl	IA		gho	
		PA		gho	
ghoh$_1$	web	IA		ghoh	
ghui$_1$	kill-12	PA		ghui	
ghui$_1$	kill-12	FA		ghelh	
ghut$_1$	saw	IA		ghut	
		PA		ghut	
		FA		ghut	
		OA		ghut	
		PN		ghut	
		FN		ghut	
		ON		ghut	
ghuz$_1$	capsize	PA		ghuz	
		FA		ghus	
		OA		ghes	
		OA		ghus	
		PN		ghus	
		FN		ghus	
gwuih$_1$	trap	PA		gwui	
		FA		gwui	
		OA		gwuih	
gwut$_1$	dull	IA	stat	gwut	
ja$_1$	say	PA		ja	
jas$_1$	fish	IA		jas	
		PA		jus	
		FA		jus	
		OA		jus	
jas̲$_4$	secrete	IA		jas̲	
		PA		jaz̲	
		FA		jas̲	
		OA		jas̲	
		PN		jaz̲	
jeh$_1$	scrape	IA		jeh	
jooh$_1$	hop	IA	cont	jooh	
		IA	cust	jooh	
		PA	mom	jok	
		PA	cont	joo	
jooh$_2$	rack	IA		jooh	
joot$_1$	diarhea	IA		joot	
		PA		joot	
jum$_1$	pop	IA		jum	
		PA		jum	
		FA		jum	
jut$_1$	decay	IA		jut	
		PA		jut	
		FA		jut	
		OA		jut	
jut$_2$	smoke moves	IA	prog	jolh	
jut$_3$	fearsome	IA		jut	
ka'$_1$	sew	IA		ka'	
		PA		kai	
		FA		ka'	
		OA		kai	
kaih$_1$	class-coc-c	IA	mom	kaih	
		IA	cont	ka	
		PA	mom	kai	
kaih$_1$	class-coc	FA	mom	kalh	
		OA	mom	kai	
		OA	mom	kalh	
		PN	mom	kal	
kak$_1$	shallow	IA	stat	kak	
kat$_1$	fall-3	IA	cont	kat	
kat$_1$	fall (3+)	PA	mom	kat	
kat$_1$	fall-3	PA	mom	kat	
kat$_1$	fall (3+)	FA	mom	kut	
ke$_1$	sit-2	IA		ke	
ke$_2$	boat	IA	cont	ke	
		IA	prog	kelh	
		IA	cust	kuih	
		PA	mom	kui	
		PA	cont	kui	
		FA	mom	kelh	
		FA	cont	kelh	1
		OA	mom	ke	
		OA	cont	ke	
		PN	cont	kel	
		FN	mom	kel	
		FN	cont	ke	
		ON	cont	ke	
ke$_3$	cranky	IA	stat	ke	
		FA	stat	ke	
ket$_1$	buy	IA		ket	
		PA		ket	
		FA		kulh	
		OA		ket	
		PN		ket	
		FN		kulh	
ket$_2$	slip	IA	stat	kut	
		IA	cont	kut	
		PA	mom	kut	
		OA	mom	ket	
		OA	mom	kut	
		FN	mom	kut	
koh$_1$	track	IA	prog	koh	
kui$_1$	tasty	IA	stat	kui	
kuk$_1$	slap	IA		kat	

– **305** –

(17)

The provision of a topical index is one of the more unusual features of these dictionaries. Reference dictionaries are almost always organized orthographically or phonologically, but it is not uncommon for dictionaries produce by communities whose language is endangered to be topically organized. This is probably due to the fact that community members regard the language as a repository of culture and so see a topical organization as a sensible reflection of their culture. From their perspective, an alphabetically arranged dictionary is in an essentially random order.

Where the topical organization is the only one provided, this is problematic, for it is very difficult to use a topical dictionary to look up an unknown word. Furthermore, topical dictionaries are usually limited to nouns, since it is nouns that are most easily organized into topical categories. Even where other syntactic categories are included, a topical organization tends to limit the comprehensiveness of the dictionary because of the difficulty of fitting many words into topical categories.

In most circumstances, therefore, a purely topical organization is not desirable. However, a topical index to a dictionary whose principal component is organized orthographically, is a useful addition. In addition to providing a facet of the dictionary that makes sense to the elders, a topical index is very useful for language teachers and curriculum developers who wish to create lessons dealing with particular topics. For similar reasons, it is useful to language learners, who can bone up on the vocabulary for a particular area. A topical index is also useful for lexicographers. If well done, it provides a good record of what has been done and shows up gaps in the coverage of the existing dictionary.

It is not terribly difficult to generate a topical index from a computer database. Indeed, I am surprised that it has not been done more often. The key is to tag each record with a semantic field specification. The records are then sorted by the semantic field specification and index entries generated in that order. Section headings are generated from the semantic field specifications by keeping track of changes and emitting a new section or subsection header whenever there is a change. Once the categories have been chosen and records tagged, the arrangement of the topical index can easily be changed by changing the ordering of the categories in the sort order specification.

For those such as myself with a propensity to classification, it is tempting to create a deep classification, one with many levels. This is unwise. It is difficult to format more than two levels of headings attractively, and a classification that is too highly ramified produces many sections with very few entries. This is not merely an aesthetic consideration — not only does the user not need an elaborate classification, but I believe that one actually makes it more difficult to find things.

We have only begun generating topical indices fairly recently, so due to our own lack of experience and, apparently, that of other lexicographers, some problems remain. One is that of redundant entries. The entries we make come from the inverse header field, which was originally intended for generating the English to Carrier headers. This field often contains multiple entries, since it is often convenient to provide more than one way to look up a word. The various inverse headers tend to be distributed through the dictionary by alphabetical sorting, but in a topical index they of course are clustered together. This looks odd and in general is probably unncessary. For example, in the page illustrated in (12), we have both “Sockeye

Salmon" and "Salmon, Sockeye", which seems unnecessary and a little bit ugly. One solution would be to create a separate topical index header field for each record and enter it manually. Another would be to use only one of the inverse headers, perhaps the first. Still another would be to generate topical index headers automatically from the gloss field, as Nathan and Austin (1992) suggest for inverse headers. I do not know whether there is a good algorithm for this.

Another problem is what classification to use. It is not too hard to devise a reasonable classification for nouns. Indeed, there are a number of models that can be used. However, it is rare for topical dictionaries to include verbs. Where they do, they tend to list those verbs that are obviously and specifically associated with the categories used for nouns and to make no attempt at listing verbs in a comprehensive fashion. If, as seems desirable, one wants to include verbs comprehensively in a topical index, it is necessary to find a good classificatory scheme for verbs.

The classification used in the most recent Carrier dictionaries represents a first attempt at this. There being little precedent known to me, it was arrived at larger by brainstorming. I found the classification of English verbs by Levin (1993) useful as a stimulus, but it was not possible to follow it closely since it is based to a considerable extent on syntactic properties that are not relevant to a topical classification. Devising a really good classification system for verbs remains a topic for research.

One thing made possible by generating dictionaries from a database is printing them in a variety of writing systems. Our system is set up to allow for printing in four different writing systems. These are exemplified in (18), which presents the words "old man" and "frequently" in all four writing systems.

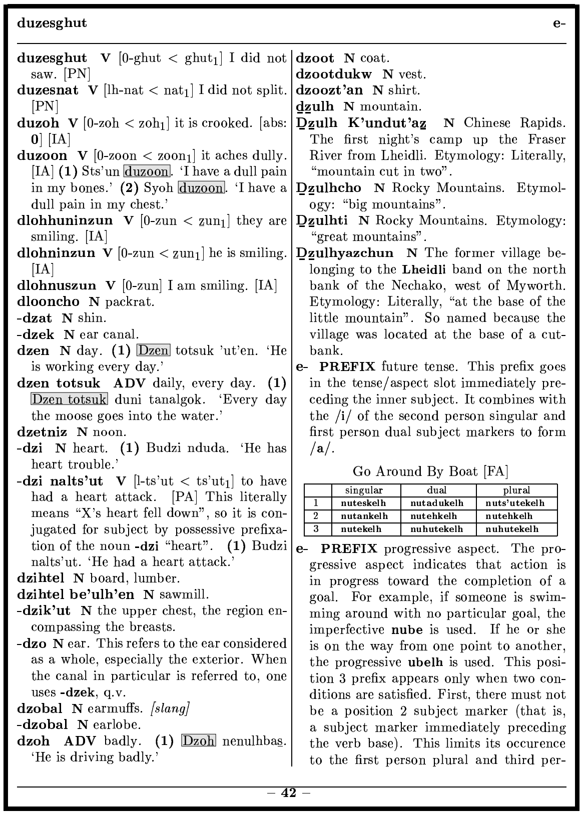
(18) Examples of the Four Available Writing Systems

Carrier Linguistic Committee	duneti	lhghun
Morice Roman	denéthi	łren
Carrier Syllabics	ᑐᑐᗪ	ᑊᐅ’
International Phonetic Alphabet	dʌneti	ɬɣʌn

The writing system in general use for Carrier is the Carrier Linguistic Committee writing system, a Roman-based system developed in the 1960s. The Carrier material in the dictionary databases is represented in this sytem, with the slight modification that underscores, used to distinguish lamino-dental fricatives and affricates from apico-alveolars, are written as separate characters preceding the consonant in the database. A typical page is shown in (19).

Some elders prefer the writing system that they learned from the 1938 edition of the Roman Catholic Prayerbook, which is the phonetic writing system used by Father Adrien-Gabriel Morice in his scholarly publications. Rather than attempting to implement allophonic rules, we generate a phonemicized version of this writing system. This is illustrated in (20).

The first writing system used for Carrier was the so-called Carrier syllabics, a derivative of the Cree syllabics introduced in 1885. It is no longer widely used, but some people strongly favor it. A page in syllabics is illustrated in (21). Finally, for the use of linguists and occasional others it is possible to produce dictionaries in the International Phonetic Alphabet as illustrated in (22).



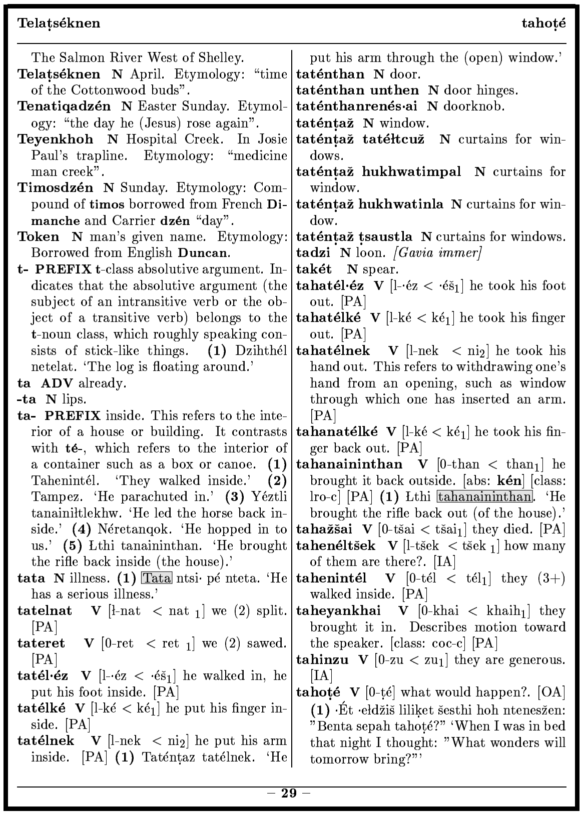
duzesghut **V** [0-ghut < ghut$_1$] I did not saw. [PN]
duzesnat **V** [lh-nat < nat$_1$] I did not split. [PN]
duzoh **V** [0-zoh < zoh$_1$] it is crooked. [abs: **0**] [IA]
duzoon **V** [0-zoon < zoon$_1$] it aches dully. [IA] **(1)** Sts'un duzoon. 'I have a dull pain in my bones.' **(2)** Syoh duzoon. 'I have a dull pain in my chest.'
dlohhuninzun **V** [0-zun < z̲un$_1$] they are smiling. [IA]
dlohninzun **V** [0-zun < z̲un$_1$] he is smiling. [IA]
dlohnuszun **V** [0-zun] I am smiling. [IA]
dlooncho **N** packrat.
-dzat **N** shin.
-dzek **N** ear canal.
dzen **N** day. **(1)** Dzen totsuk 'ut'en. 'He is working every day.'
dzen totsuk **ADV** daily, every day. **(1)** Dzen totsuk duni tanalgok. 'Every day the moose goes into the water.'
dzetniz **N** noon.
-dzi **N** heart. **(1)** Budzi nduda. 'He has heart trouble.'
-dzi nalts'ut **V** [l-ts'ut < ts'ut$_1$] to have had a heart attack. [PA] This literally means "X's heart fell down", so it is conjugated for subject by possessive prefixation of the noun **-dzi** "heart". **(1)** Budzi nalts'ut. 'He had a heart attack.'
dzihtel **N** board, lumber.
dzihtel be'ulh'en **N** sawmill.
-dzik'ut **N** the upper chest, the region encompassing the breasts.
-dzo **N** ear. This refers to the ear considered as a whole, especially the exterior. When the canal in particular is referred to, one uses **-dzek**, q.v.
dzobal **N** earmuffs. *[slang]*
-dzobal **N** earlobe.
dzoh **ADV** badly. **(1)** Dzoh nenulhbas̲. 'He is driving badly.'
dzoot **N** coat.
dzootdukw **N** vest.
dzoozt'an **N** shirt.
d̲z̲ulh **N** mountain.
D̲z̲ulh K'undut'az̲ **N** Chinese Rapids. The first night's camp up the Fraser River from Lheidli. Etymology: Literally, "mountain cut in two".
D̲z̲ulhcho **N** Rocky Mountains. Etymology: "big mountains".
D̲z̲ulhti **N** Rocky Mountains. Etymology: "great mountains".
D̲z̲ulhyazchun **N** The former village belonging to the **Lheidli** band on the north bank of the Nechako, west of Myworth. Etymology: Literally, "at the base of the little mountain". So named because the village was located at the base of a cutbank.
e- **PREFIX** future tense. This prefix goes in the tense/aspect slot immediately preceding the inner subject. It combines with the /i/ of the second person singular and first person dual subject markers to form /**a**/.

Go Around By Boat [FA]

	singular	dual	plural
1	**nuteskelh**	**nutadukelh**	**nuts'utekelh**
2	**nutankelh**	**nutehkelh**	**nutehkelh**
3	**nutekelh**	**nuhutekelh**	**nuhutekelh**

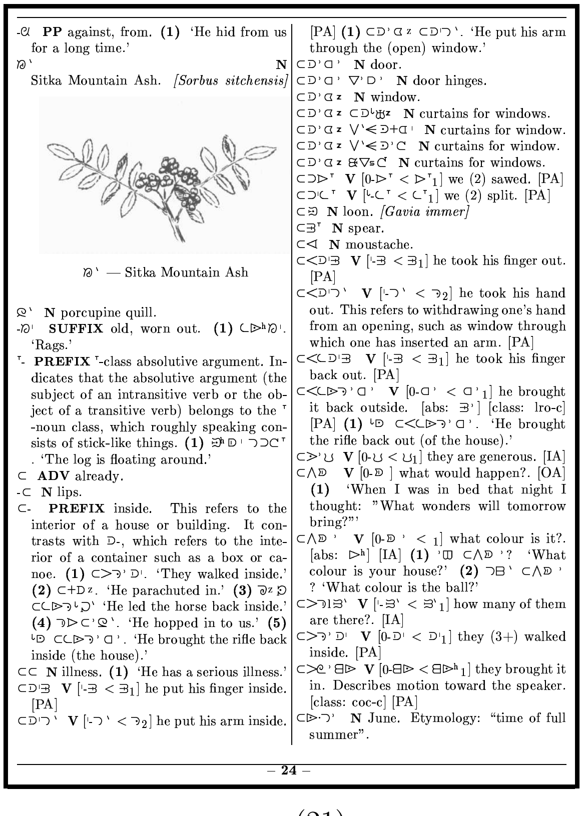
e- **PREFIX** progressive aspect. The progressive aspect indicates that action is in progress toward the completion of a goal. For example, if someone is swimming around with no particular goal, the imperfective **nube** is used. If he or she is on the way from one point to another, the progressive **ubelh** is used. This position 3 prefix appears only when two conditions are satisfied. First, there must not be a position 2 subject marker (that is, a subject marker immediately preceding the verb base). This limits its occurence to the first person plural and third per-

(19)



The Salmon River West of Shelley.
Telaṭséknen **N** April. Etymology: "time of the Cottonwood buds".
Tenatiqadzén **N** Easter Sunday. Etymology: "the day he (Jesus) rose again".
Teyenkhoh **N** Hospital Creek. In Josie Paul's trapline. Etymology: "medicine man creek".
Timosdzén **N** Sunday. Etymology: Compound of **timos** borrowed from French **Dimanche** and Carrier **dzén** "day".
Token **N** man's given name. Etymology: Borrowed from English **Duncan**.
t- **PREFIX** **t**-class absolutive argument. Indicates that the absolutive argument (the subject of an intransitive verb or the object of a transitive verb) belongs to the **t**-noun class, which roughly speaking consists of stick-like things. **(1)** Dzihthél netelat. 'The log is floating around.'
ta **ADV** already.
-ta **N** lips.
ta- **PREFIX** inside. This refers to the interior of a house or building. It contrasts with **té-**, which refers to the interior of a container such as a box or canoe. **(1)** Tahenintél. 'They walked inside.' **(2)** Tampez. 'He parachuted in.' **(3)** Yéztli tanainiłtlekhw. 'He led the horse back inside.' **(4)** Néretanqok. 'He hopped in to us.' **(5)** Łthi tanaininthan. 'He brought the rifle back inside (the house).'
tata **N** illness. **(1)** Tata ntsi· pé nteta. 'He has a serious illness.'
tatelnat **V** [ł-nat < nat $_1$] we (2) split. [PA]
tateret **V** [0-ret < ret $_1$] we (2) sawed. [PA]
tatél·éz **V** [l-·éz < ·éš$_1$] he walked in, he put his foot inside. [PA]
tatélké **V** [l-ké < ké$_1$] he put his finger inside. [PA]
tatélnek **V** [l-nek < ni$_2$] he put his arm inside. [PA] **(1)** Taténṭaz tatélnek. 'He put his arm through the (open) window.'
taténthan **N** door.
taténthan unthen **N** door hinges.
taténthanrenés·ai **N** doorknob.
taténṭaž **N** window.
taténṭaž tatéłtcuž **N** curtains for windows.
taténṭaž hukhwatimpal **N** curtains for window.
taténṭaž hukhwatinla **N** curtains for window.
taténṭaž ṭsaustla **N** curtains for windows.
tadzi **N** loon. *[Gavia immer]*
takét **N** spear.
tahatél·éz **V** [l-·éz < ·éš$_1$] he took his foot out. [PA]
tahatélké **V** [l-ké < ké$_1$] he took his finger out. [PA]
tahatélnek **V** [l-nek < ni$_2$] he took his hand out. This refers to withdrawing one's hand from an opening, such as window through which one has inserted an arm. [PA]
tahanatélké **V** [l-ké < ké$_1$] he took his finger back out. [PA]
tahanaininthan **V** [0-than < than$_1$] he brought it back outside. [abs: **kén**] [class: lro-c] [PA] **(1)** Łthi tahanaininthan. 'He brought the rifle back out (of the house).'
tahažšai **V** [0-tšai < tšai$_1$] they died. [PA]
taheneltšek **V** [l-tšek < tšek $_1$] how many of them are there?. [IA]
tahenintél **V** [0-tél < tél$_1$] they (3+) walked inside. [PA]
taheyankhai **V** [0-khai < khaih$_1$] they brought it in. Describes motion toward the speaker. [class: coc-c] [PA]
tahinzu **V** [0-zu < zu$_1$] they are generous. [IA]
tahoṭé **V** [0-ṭé] what would happen?. [OA] **(1)** ·Ét ·ełdžiš liliḳet šesthi hoh ntenesžen: "Benta sepah tahoṭé?" 'When I was in bed that night I thought: "What wonders will tomorrow bring?"'

(20)

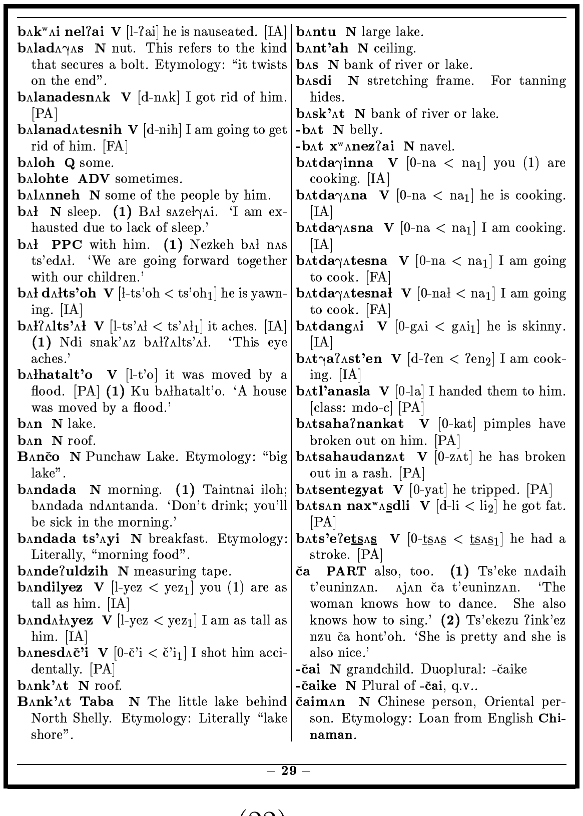


-ᗧ **PP** against, from. **(1)** 'He hid from us for a long time.'
ᘐ` **N** Sitka Mountain Ash. *[Sorbus sitchensis]*

ᘐ` — Sitka Mountain Ash

ᘓ` **N** porcupine quill.
-ᘐ' **SUFFIX** old, worn out. **(1)** ᑕᐅʰᘐ'. 'Rags.'
ᐪ- **PREFIX** ᐪ-class absolutive argument. Indicates that the absolutive argument (the subject of an intransitive verb or the object of a transitive verb) belongs to the ᐪ -noun class, which roughly speaking consists of stick-like things. **(1)** ᗱʰᗜ' ᑐᑐᑕᐪ . 'The log is floating around.'
ᑕ **ADV** already.
-ᑕ **N** lips.
ᑕ- **PREFIX** inside. This refers to the interior of a house or building. It contrasts with ᗜ-, which refers to the interior of a container such as a box or canoe. **(1)** ᑕᐳᑐ'ᗜ'. 'They walked inside.' **(2)** ᑕ+ᗜᶻ. 'He parachuted in.' **(3)** ᑐᶻᗜ ᑕᑕᐅᑐᒡᗜ` 'He led the horse back inside.' **(4)** ᑐᐅᑕ'ᘓ`. 'He hopped in to us.' **(5)** ᒡᗜ ᑕᑕᐅᑐ'ᗡ'. 'He brought the rifle back inside (the house).'
ᑕᑕ **N** illness. **(1)** 'He has a serious illness.'
ᑕᗜ'ᗱ **V** ['-ᗱ < ᗱ$_1$] he put his finger inside. [PA]
ᑕᗜ'ᑐ` **V** ['-ᑐ` < ᑐ$_2$] he put his arm inside. [PA] **(1)** ᑕᗜ'ᗡᶻ ᑕᗜ'ᑐ`. 'He put his arm through the (open) window.'
ᑕᗜ'ᗡ' **N** door.
ᑕᗜ'ᗡ' ᐁ'ᗜ' **N** door hinges.
ᑕᗜ'ᗡᶻ **N** window.
ᑕᗜ'ᗡᶻ ᑕᗜᒡᗾᶻ **N** curtains for windows.
ᑕᗜ'ᗡᶻ ᐯ`ᗕᗜ+ᗡ' **N** curtains for window.
ᑕᗜ'ᗡᶻ ᐯ`ᗕᗜ'ᑕ **N** curtains for window.
ᑕᗜ'ᗡᶻ ᗸᐁᔆᑕ **N** curtains for windows.
ᑕᑐᐅᐪ **V** [0-ᐅᐪ < ᐅᐪ$_1$] we (2) sawed. [PA]
ᑕᑐ'ᑕᐪ **V** [ᒡ-ᑕᐪ < ᑕᐪ$_1$] we (2) split. [PA]
ᑕᗦ **N** loon. *[Gavia immer]*
ᑕᗱᐪ **N** spear.
ᑕᐊ **N** moustache.
ᑕᐸᗜ'ᗱ **V** ['-ᗱ < ᗱ$_1$] he took his finger out. [PA]
ᑕᐸᗜ'ᑐ` **V** ['-ᑐ` < ᑐ$_2$] he took his hand out. This refers to withdrawing one's hand from an opening, such as window through which one has inserted an arm. [PA]
ᑕᐸᑕᗜ'ᗱ **V** ['-ᗱ < ᗱ$_1$] he took his finger back out. [PA]
ᑕᐸᑕᐅᑐ'ᗡ' **V** [0-ᗡ' < ᗡ'$_1$] he brought it back outside. [abs: ᗱ'] [class: lro-c] [PA] **(1)** ᒡᗜ ᑕᐸᑕᐅᑐ'ᗡ'. 'He brought the rifle back out (of the house).'
ᑕᐳ'ᑌ **V** [0-ᑌ < ᑌ$_1$] they are generous. [IA]
ᑕᐱᗜ **V** [0-ᗜ] what would happen?. [OA] **(1)** 'When I was in bed that night I thought: "What wonders will tomorrow bring?"'
ᑕᐱᗜ ' **V** [0-ᗜ ' < $_1$] what colour is it?. [abs: ᐅʰ] [IA] **(1)** 'ᗯ ᑕᐱᗜ '? 'What colour is your house?' **(2)** ᑐᗷ` ᑕᐱᗜ ' ? 'What colour is the ball?'
ᑕᐳᑐlᗱ` **V** ['-ᗱ` < ᗱ`$_1$] how many of them are there?. [IA]
ᑕᐳᑐ'ᗜ' **V** [0-ᗜ' < ᗜ'$_1$] they (3+) walked inside. [PA]
ᑕᐳᘓ'ᗸᐅ **V** [0-ᗸᐅ < ᗸᐅʰ$_1$] they brought it in. Describes motion toward the speaker. [class: coc-c] [PA]
ᑕᐅ·ᑐ' **N** June. Etymology: "time of full summer".

(21)



bʌkʷʌi nelʔai **V** [l-ʔai] he is nauseated. [IA]
bʌladʌγʌs **N** nut. This refers to the kind that secures a bolt. Etymology: "it twists on the end".
bʌlanadesnʌk **V** [d-nʌk] I got rid of him. [PA]
bʌlanadʌtesnih **V** [d-nih] I am going to get rid of him. [FA]
bʌloh **Q** some.
bʌlohte **ADV** sometimes.
bʌlʌnneh **N** some of the people by him.
bʌł **N** sleep. **(1)** Bʌł sʌzełγʌi. 'I am exhausted due to lack of sleep.'
bʌł **PPC** with him. **(1)** Nezkeh bʌł nʌs ts'edʌł. 'We are going forward together with our children.'
bʌł dʌłts'oh **V** [ł-ts'oh < ts'oh$_1$] he is yawning. [IA]
bʌłʔʌlts'ʌł **V** [l-ts'ʌł < ts'ʌł$_1$] it aches. [IA] **(1)** Ndi snak'ʌz bʌłʔʌlts'ʌł. 'This eye aches.'
bʌłhatalt'o **V** [l-t'o] it was moved by a flood. [PA] **(1)** Ku bʌłhatalt'o. 'A house was moved by a flood.'
bʌn **N** lake.
bʌn **N** roof.
Bʌnčo **N** Punchaw Lake. Etymology: "big lake".
bʌndada **N** morning. **(1)** Taintnai iloh; bʌndada ndʌntanda. 'Don't drink; you'll be sick in the morning.'
bʌndada ts'ʌyi **N** breakfast. Etymology: Literally, "morning food".
bʌndeʔuldzih **N** measuring tape.
bʌndilyez **V** [l-yez < yez$_1$] you (1) are as tall as him. [IA]
bʌndʌłʌyez **V** [l-yez < yez$_1$] I am as tall as him. [IA]
bʌnesdʌč'i **V** [0-č'i < č'i$_1$] I shot him accidentally. [PA]
bʌnk'ʌt **N** roof.
Bʌnk'ʌt Taba **N** The little lake behind North Shelly. Etymology: Literally "lake shore".
bʌntu **N** large lake.
bʌnt'ah **N** ceiling.
bʌs **N** bank of river or lake.
bʌsdi **N** stretching frame. For tanning hides.
bʌsk'ʌt **N** bank of river or lake.
-bʌt **N** belly.
-bʌt xʷʌnezʔai **N** navel.
bʌtdaγinna **V** [0-na < na$_1$] you (1) are cooking. [IA]
bʌtdaγʌna **V** [0-na < na$_1$] he is cooking. [IA]
bʌtdaγʌsna **V** [0-na < na$_1$] I am cooking. [IA]
bʌtdaγʌtesna **V** [0-na < na$_1$] I am going to cook. [FA]
bʌtdaγʌtesnał **V** [0-nał < na$_1$] I am going to cook. [FA]
bʌtdangʌi **V** [0-gʌi < gʌi$_1$] he is skinny. [IA]
bʌtγaʔʌst'en **V** [d-ʔen < ʔen$_2$] I am cooking. [IA]
bʌtl'anasla **V** [0-la] I handed them to him. [class: mdo-c] [PA]
bʌtsahaʔnankat **V** [0-kat] pimples have broken out on him. [PA]
bʌtsahaudanzʌt **V** [0-zʌt] he has broken out in a rash. [PA]
bʌtsentez̲yat **V** [0-yat] he tripped. [PA]
bʌtsʌn naxʷʌs̲dli **V** [d-li < li$_2$] he got fat. [PA]
bʌts'eʔet̲s̲ʌs̲ **V** [0-t̲s̲ʌs̲ < t̲s̲ʌs̲$_1$] he had a stroke. [PA]
ča **PART** also, too. **(1)** Ts'eke nʌdaih t'euninzʌn. ʌjʌn ča t'euninzʌn. 'The woman knows how to dance. She also knows how to sing.' **(2)** Ts'ekezu ʔink'ez nzu ča hont'oh. 'She is pretty and she is also nice.'
-čai **N** grandchild. Duoplural: -čaike
-čaike **N** Plural of **-čai**, q.v..
čaimʌn **N** Chinese person, Oriental person. Etymology: Loan from English **Chinaman**.

(22)

5. Generating Printed Dictionaries

Printed dictionaries are generated from the databases by executing programs that extract information from the database, sort it, and generate a set of files suitable for input to the TeX formatter. The TeX program is then executed on a master format file, which incorporates the various files generated from the database along with various fixed files, such as the explanation of the writing system. The overall process is illustrated in (24).

Since each dictionary has so many pieces, and there are so many intermediate stages, the production of a dictionary is quite complex. Not only would it be difficult to keep track of by hand, but if an error occurs at some point, it is desirable not to have to regenerate everything from scratch, but only what is necessary. The generation of a dictionary is therefore controlled by the *make* program, a standard UNIX utility originally designed for controlling the compilation of computer programs. Where there are no errors, generation of a dictionary requires five commands, exemplified in (23):[6]

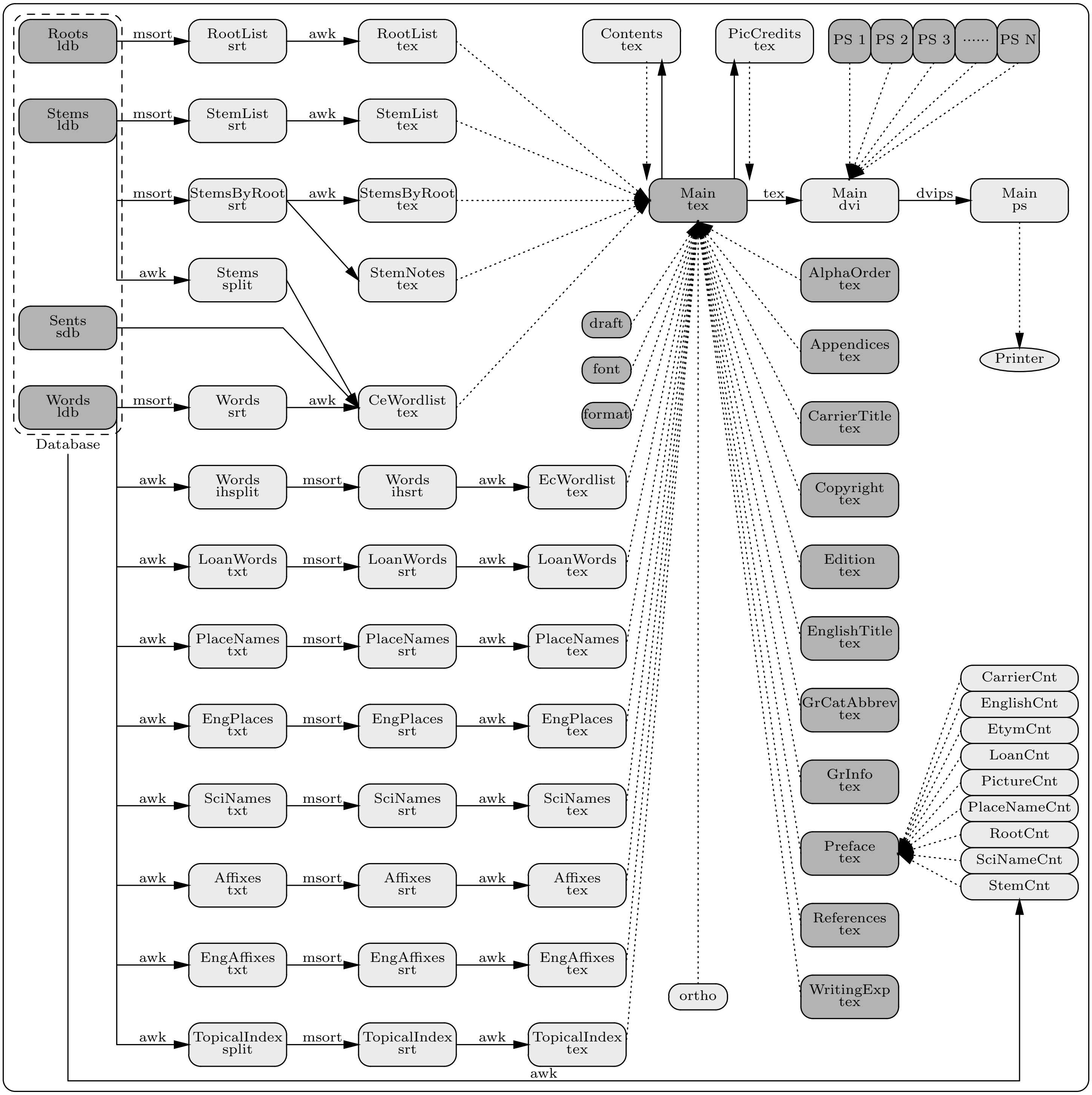
(23) Commands for Generating a Dictionary

```
01 make CLCDict
02 cd Forms
03 make
04 make
05 dvips -o Main.ps Main.dvi
```

The first line tells the *make* program to do whatever is necessary to generate the target *CLCDict*. It will look for instructions as to how to do this as explained below. Assuming that there are no errors, the result of this step will be the generation of the various files that go into the dictionary. The second line changes the directory to a subdirectory called *Forms* in which the master format file is kept. The third line executes the *make* program again. Since it is called with no argument, it will attempt to create the first target specified in the *makefile*. If there are no errors, the result of this step is the execution of TeX and the generation of a device-independent printer language file. The second *make* command, on line 04, is not an error. It is executed twice in order to make sure that the page numbers in the table of contents and figure credits are correct. Since page numbers are determined anew each time TeX is run, information containing page numbers must be written out during one run of TeX, then read back in and used during a subsequent run. The *dvi* file is then converted by the *dvips* program into the Postscript printer language. It is at this stage that images are actually incorporated. The Postscript file may then be viewed on the computer or sent to a printer.[7]

[6] Here as in subsequent examples, the line numbers have been added to facilitate exposition. They are not part of the makefile or program.

[7] In practice, when preparing to print rather than to examine the outout on a computer monitor, *dvips* will generally be run more than once, each time generating a Postscript file containing a specified section of the dictionary. Sending a large document to the printer in chunks of, say, fifty pages each, is better than sending it all in one piece. It allows other users to get their jobs



Dark boxes indicate underived files. Light boxes indicate derived files. Dotted lines indicate inclusion. Solid lines indicate derivation.

(24) Generating a Printed Dictionary

in, provides opportunities to let the printer rest and make adjustments, and permits minor changes to be made in sections that have not yet been printed.

The *make* program reads a file called a *makefile* that contains specifications of dependencies among files and of how files may be created. Here is an extract from the makefile for a Carrier dictionary.

(25) Part of a Make File

```
01 CLCDict:      CLCCeWL.tex CLCEcWordlist.tex CLCScientificNames.tex
                 CLCLoanWords.tex CLCPlaceNames.tex CLCRootList.tex
                 CLCEnglishRootList.tex CLCStemList.tex CLCStemsByRoot.tex
                 NotesOnStems.tex CLCAffixList.tex CLCAffixesByGloss.tex
                 CLCEnglishPlaceNames.tex CLCEcTopicalIndex.tex
02
03 CLCCeWL.tex:  Ce.srt StemList.sp
04               gawk -f $CAR/Scripts/PrepDict.awk -v Orth=clc -v Version=scholars
                 Ce.srt > tmp
05               cat GenInfo.tex tmp > CLCCeWL.tex
06               echo "\ortho=1" > Forms/ortho
07               rm tmp
08
09 Ce.srt:       NoComments.txt
10               msort -t "^P:" -c l -s ${CAR}/orders/carriernc.ord -x exclusions
                       -t "^P:" -c l -s ${CAR}/orders/carrier2.ord
                       -t "^C:" -c l -s ${CAR}/orders/cats.ord
                       -t "^G:" -o -s ${CAR}/orders/english.ord
                       -d % NoComments.txt > Ce.srt
```

The first line indicates that the target CLCDict depends on the files CLCEWL.tex, CLCEcWordlist.tex, etc. This target is not actually a file but is used to tell the *make* program what kind of dictionary we want to generate. We would do this by giving the command `make CLCDict`. If the files on which CLCDict depends do not exist or are out of date, the program will attempt to create them.

Line 03 says that one of the files needed, *CLCCeWL.tex*, depends on two other files, *Ce.srt* and *Stemlist.sp.* Lines 04 through 07 give the commands that must be executed in order to create *CLCCeWL.tex* from these two files. Line 04 runs an AWK program, whose main output goes into the file called *tmp.* The file *GenInfo.tex* is also created as a side effect. Line 05 combines the two files. Line 06 creates a little file that transmits information to TEXabout the writing system in use, and line 07 deletes the temporary file *tmp.*

Line 09 tells us that the file *Ce.srt* depends on *NoComments.txt.* Line 10 says that it may be created by running the *msort* program. The rather elaborate tell msort to sort first on the pronounciation field, using a sort order specified in a file and exlucing from consideration characters specified in another file, then to subsort (breaking ties) on the same field using a different sort order and with no exclusions, then if necessary to subsort on the gloss and finally if necessary to subsort on the category field.

The bulk of the work of extracting information from the database and formatting it is done by small programs written in AWK (Aho, Kernighan, and Weinberger 1988). Several features of AWK make it particularly suitable for this kind of text processing:

1. automatic storage allocation;

2. associative arrays, that is, arrays whose indices can be strings rather than integers;
3. regular expression matching and substitution;
4. automatic parsing of input into records, and of records into fields;

A small AWK program is illustrated in (26). This is a slightly simplified version of the program that extracts records containing semantic field specifications for further processing that ultimately generates a topical index.

(26) An AWK Program

```
01  #Extract records for topical index, that is, records
02  #containing an SF field, an IH field, and a G field.
03  BEGIN {
04     RS = "";
05     FS = "\n?%";
06  }
07  {
08     #Create an associative array with tags as indices.
09     for(i = 2; i <= NF; i++) {
10        split($i, f, ":");
11        rec[f[1]]=substr($i,index($i,":")+1);
12     }
13     if(("SF" in rec) && ("G" in rec) && ("IH" in rec)){
14        printf("%s\n\n",$0);
15     }
16     CleanUp();
17  }
```

The program begins with a BEGIN pattern, which is automatically executed at the beginning of the program, no matter what the input. Two actions are taken. The record separator RS is set to a value that makes it treat blocks of text separated by blank lines as records. The field separator FS is set to a percent sign preceded by at least one newline character. (Using the percent sign as the basic field separator allows fields to span more than one line. However, since a percent sign might be part of the value of a field, we treat as field separators only those percent signs preceded by newlines.

The remainder of the program consists of a single pattern-action pair in which the pattern is empty, so the action is taken for each new record. Line 09 loops through the fields that AWK automatically parses out of the current record. It starts with field number two because all of our fields start with a percent-sign field separator, so each record in effect begins with an empty field. AWK numbers the fields $1, $2, etc. The number of fields parsed out is put into the variable NF.

In line 10 the field is split into pieces separated by colons and these pieces are put into the array “f”. f[1] therefore contains the tag of the first field. In line 11 the contents of the field are put into the array “rec”. The index is the tag, which in line 10 was stored in f[1]. The part of the line following the equal sign removes everything preceding the first colon.[8] As of line 13, therefore, the array "rec" contains the fields of the record, now indexed by their tags.[9]

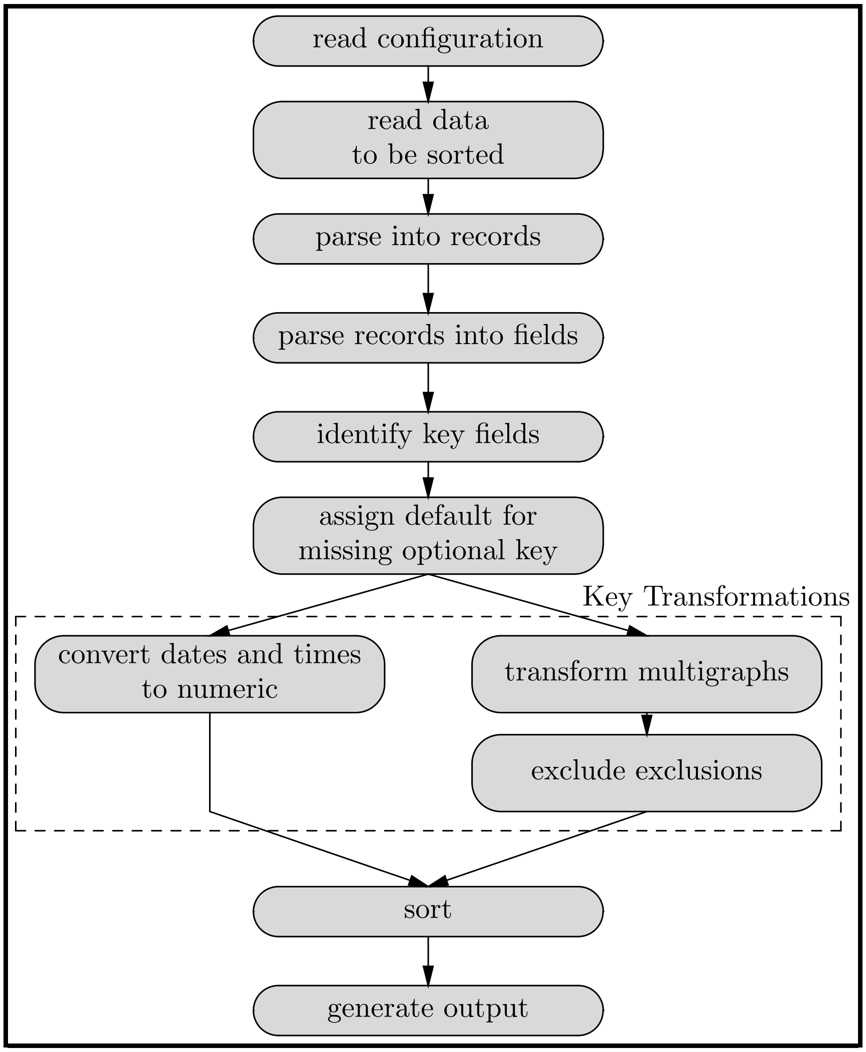
Line 13 tests whether the record contains a semantic field specification, a gloss, and an inverse header. If so, line 14 prints the entire record ($0), followed by two blank lines to separate it from the next record. Line 16 calls a function not shown here that cleans up in preparation for the next record. For example, it needs to delete the current contents of the array "rec". If it did not, fields left over from previous records would be mixed with those for the current record.

As can be seen from the diagram in (24), there are numerous points at which information must be sorted into the proper order. These sorts are carried out by a program called *msort*, described in detail in Poser (2000). This is a program for sorting text files in sophisticated ways, intended especially for linguistic databases. It allows arbitrary sort orders to be specified, with ranks defined for large numbers of multigraphs of effectively unlimited length. Records need not be single lines of text but may be delimited in a number of ways. The entire record may used as the sort key, or a particular field may be used. Key fields may be selected either by position in the record or by matching a regular expression to a tag. *msort* is capable of sorting on several keys, so that when two records tie on one key, the tie may be broken on another. Each key may have its own sort order. Any or all keys may be optional. In addition to lexicographic sorting, sorting by numerical value, date or time is supported. For each key a distinct set of characters may be excluded from consideration when sorting in any combination of initial, final, and medial position in the key field.

[8] If we knew that the colon separating the tag from the contents of the field were the only one, we could just use f[2], but it is quite possible that there will be colons within the content of the field, so we can't assume that f[2] will contain all of the content.

[9] Notice that if two fields have the same tag, the last one in the record will be the only one in "rec". For most purposes this is alright, but a different and more complex parsing technique must be used when we want to extract repeated fields.

(27) The Sort Process



A good example of the utility of a sorting program with capabilities like those of *msort* is provided by the sorting needed to generate the table of stems sorted by roots. The relevant portion of the make file, slightly edited for presentation, is given in (28).

(28) The Makefile Entry for Sorting Stems by Roots

```
StemsByRoot.srt:  StemList.ldb
                  -msort -d %  -t "^ROOT" -c l -s ${CAR}/orders/carrier1.ord
                               -t "^RID:" -c n
                               -t "^TM:" -c l -s ${CAR}/orders/TM.ord
                               -t "^ASP" -c l -s ${CAR}/orders/ASP.ord
                               -t "^STEM:" -c l -s ${CAR}/orders/carrier1.ord
                               < StemList.ldb > CLCStemsByRoot.srt
```

(29) The Sort Order File for Aspect Categories

```
stat    stative
mom     momantaneous
cont    continuous
dist    distributive
prog    progressive
sem     semelfactive
rep     repetitive
cust    customary
```

This makes use of the ability to specify arbitrary sort orders that have nothing to do with any language's alphabetical order as well as the ability to handle long multigraphs. Here both abbreviated and full aspect names are provided for. The longest is 13 characters long.

A similar use of this ability is in putting the topical index into order. Here, the records must first be sorted by semantic field, then, within their category, alphabetically. Semantic field specifications may be quite long; the longest, at present, is *gathering-plants-scrapingcambium*, which is 32 characters.

The output of the programs that extract information from the database and format it consits of pieces of text formatted in TeX. The final step in printing the dictionary is to execute TeX on a file that contains the layout for the dictionary and, at appropriate places, instructions to insert the contents of other files, containing the actual dictionary information, which have been generated from the database. The bit of TeX-formatted text underlying the first entry on in the Carrier-English page in (10) is shown in (30). The macro *\hipu* sets up a hanging-indented paragraph whose first line is unindented with respect to the left margin. Lines 02, 04, and 05 print the headword, category abbreviation, and gloss, which in this case is a cross-reference to the singular. The macros *\pft*, *\cft*, and *\gft* select the appropriate fonts. Line 03 does not directly print anything, but records the information that TeX uses to keep track of the first and last entries on the page so as to generate the left and right page headers.

(30) An Entry from CLCCeWL.tex

```
01  \hipu
02  {\pft -chaike}
03  {\mark -chaike}
04  {\cft N}
05  {\gft Plural of {\qc -chai}, q.v.}.
```

6. Direct Use of the Database

Generating printed dictionaries is not the only use for the Carrier lexical databases. Another use is as on-line dictionaries. They are easily searched, by means of the search functions in Emacs and by means of other programs, such as AWK. In most respects, such an on-line lexicon is faster and more convenient to use than a printed version. I myself almost never use the printed dictionaries that I create — it is usually more convenient for me to look up anything I do not know in the computer files.

Another important use of the database is for linguistic research. An on-line lexicon can be searched in many different ways, for various combinations of information. I frequently use the regular-expression matching facilities of Emacs and of AWK to look for examples of various types.

For example, here is the EMACS Lisp code that creates a function that searches the database for Carrier words good for inclusion in critical regions of sentences elicited for the instrumental study of tone and intonation. It defines a function that uses the existing incremental regular expression search facility to find records whose headword contains only sonorants and voiced stops, sounds that do not significantly perturb the fundamental frequency contour.

(31) EMACS Lisp Code Defining a Search Function

```
(defun find-goodf0 ()
   "Find the next word good for F0."
   (interactive)
   (re-search-forward goodf0-regexp)
)
(defconst goodf0-regexp "%P:[aiueomnylbdg]+$"
   "Regular expression for recognizing words good for F0")
```

7. How and Why it Got to be That Way

This database system came about for a number of reasons. It origiinally ran on what by current standards were very slow machines, under DOS. I had no funding with which to buy specialized software or to hire other people to do part of the work or provide technical support. The better, faster machines to which I had access, though initially not at home or in the field, ran UNIX, so compatibility with UNIX systems was desirable. Since I was thoroughly familiar with *awk*, *make*, and other UNIX utilities and had extensive programming experience, and versions of these were available for DOS at little or no cost, it made sense to use them.

A second factor was the limitations of available software. Most databases were not designed for linguistic work and were not very flexible. They often required fields to be of fixed length, could not generate output in appropriate formats, and lacked the ability to sort in sufficiently sophisticated ways. I wrote *msort* because there was no truly comparable program. The standard UNIX sorting utility lacks many of the capabilities of *msort*, as do most if not all other stand-alone sorting programs of which I am aware. Sorting of comparable sophistication was and is available only within some databases, and even then, to my knowledge not all of *msort*'s capabilities are available.

At the time, the only turnkey lexical database system of which I was aware was the Summer Institute of Linguistics' Shoebox program. Shoebox is in some ways not unlike my own system but provides a more structured interface and a higher degree of integrity checking. However, Shoebox did not have the sorting capability I needed and could not easily be made to generate output in the format I desired.

The remaining approach was to use Robert Hsu's *Lexware* system, a collection of programs together with a file format that has been used to produce quite a few dictionaries. Hsu has made great contributions both to the production of individual dictionaries and to thinking about the automated production of dictionaries,[10] but his system was not quite what I wanted. For one thing, it is not a turnkey system, but requires the user to rely on Hsu to adapt his programs to their needs. Using the UNIX utilities I could just as easily have complete control and flexibility.

Yet another factor was that I did not set out, at the outset, to generate dictionaries. I initially created the database as a way of recording data I had collected and being able to search it more efficiently than I could by reading through my handwritten notes. Beyond this I had no specific plans for the database.

Finally, at the outset I did not know very much about the language, and therefore did not know exactly what sorts of information would need to be separated and what structure the database ought to have. It was important to avoid committments to details of database structure that I might later regret.

8. Discussion

All in all, this system has served my needs well. It cost nothing, was not difficult to set up, and has proved to be easy to modify and extend. The work necessary to create it may have been greater than it would have been with an existing database, but that is not clear, since few databases were well adapted to the same purposes. Now that the system is set up, though, this disadvantage is not so great. I have found it to be generally quite easy to modify and extend the existing system. Numerous changes have been made in the format of printed dictionaries without difficulty and various indices and new displays of information added. Adapting this dictionary system to Flathead and Kootenai was a simple matter. Similarly, adapting the software that generates TEX to generate HTML instead was quite simple. Once the framework is in place, many changes are easy to make.[11]

A more significant disadvantage is that it does not provide the same degree of integrity control as do most standard databases. That is, it is relatively easy for a record to become ill-formed or for records to be deleted accidentally. As I am a skilled user of the system, the integrity checks and backups provided are probably sufficient.

[10] Hsu (1994) is a valuable discussion derived from his extensive experience in this area.

[11] AWK has some limitations as a programming language. Were I to set up such a system again from scratch, I might choose a different programming language. Many people now use PERL (Wall & Randal 1991) for similar purposes. I have considered this, but to my taste PERL is not cleanly structured and suffers from kitchen sinkism. Should I decide to use a language other than AWK, it will probably be Python (Lutz 1996).

For a single expert user such as myself, probably the greatest limitation at present is the need to create and maintain relations, such as that between records for forms and example sentences, manually.

Generally speaking, this system is quite adequate for my own use and for use by similarly skilled users working one at a time. The system is not well adapted to use by more than one person or by inexpert users. Inexpert users would benefit from a system that provides more elaborate integrity checks, with an interface that carefully controlls the ways in which they can modify the database. A forms-based interface would also make it easier for inexpert users to enter information. The system is not well adapted to use by multiple users because it does not provide file or record locking, different levels of access by different users, or automatic recording of what modifications are made by whom.

References

Aho, Alfred V., Kernighan, Brian W., and Peter J. Weinberger (1988) *The AWK Programming Language.* Reading, Massachusetts: Addison-Wesley.

Amith, Jonathan (2002a) *Diccionario del idioma Nahuatl de Oapan y Ameyaltepec.* Philadelphia: the author.

Amith, Jonathan (2002b) *Dictionary of Nahuatl of Opan and Ameyaltepec.* Philadelphia: the author.

Feldman, S. I. (1986) "Make — A Program for Maintaining Computer Programs," Bell Laboratories Technical Report.

Hsu, Robert (1994) *Methods of Language Data Processing.* ms. University of Hawaii.

Knuth, Donald E. (1984) *The TEXbook.* Reading, Massachusetts: Addison-Wesley.

Levin, Beth (1993) *English Verb Classes and Alternations: A Preliminary Investigation.* Chicago: The University of Chicago Press.

Lutz, Mark (1996) *Programming Python.* Cambridge: O'Reilly and Associates.

Nathan, David and Peter Austin (1992) "Finderlists, Computer-Generated, for Bilingual Dictionaries," *International Journal of Lexicography* **5**.1.

Poser, William J. (2000) *MSORT Reference Manual*. The current version may be downloaded from `http://www.ling.upenn.edu/~wjposer/`.

Stallman, Richard M. (1984). "EMACS: The Extensible, Customizable, Self-Documenting Display Editor," *Interactive Programming Environments*. edited by D. R. Barstow, H. E. Shrobe, & E. Sandwell, (New York: McGraw-Hill), pp.300-325.

Thomason, Sarah G. (1994) *A Dictionary of Montana Salish.* Ms., University of Pittsburgh and Confederated Salish/Kootenay Tribes.

Wall, Larry and Randal L. Schwartz (1991) *Programming perl.* Cambridge: O'Reilly and Associates.