A two-level implementation for Lushootseed morphology

Deryle Lonsdale
BYU Linguistics Department

This paper describes the implementation of a computer system that processes Lushootseed word forms. Built on the two-level model and leveraging finite-state technology, the system is able both to parse surface forms to arrive at the underlying morphemic decomposition, and to generate surface forms given such a representation. Components of the system are discussed including the lexicon architecture, rule formulation and specification, and the word-structure grammar. Issues relevant to processing Lushootseed are discussed including reduplication, allomorphic variation, inflectional and derivational affixation, and morphophonemic alternations.

1 Introduction

This paper presents a computer system that performs morphological processing for Lushootseed, a Salishan language. As background, various aspects of the language's morphology are first surveyed. Then the underlying two-level approach is discussed and related to morphological properties of the language. The development of associated knowledge sources and their incorporation into the model is then described. Mention is made of particularly interesting linguistic issues such as reduplication, allomorphic variation, inflectional and derivational affixation, and morphophonemics.

The system's current level of functionality is then discussed. Examples are given that illustrate how the system operates, the types of results it obtains, and its current coverage. Other possible applications are then mentioned, plans for further development of the system are sketched, and future applications are suggested.

2 Lushootseed

This section gives a brief overview of relevant aspects of the Lushootseed language, focusing particularly on its morphology. Other aspects relevant to this paper are also touched on. The information has been gathered from the valuable and excellent grammars, readers, and dictionaries that are familiar to Lushootseed students and researchers (Hess 1976; Hess and Hilbert. 1977; Bates, Hess, and Hilbert. 1994; Hess 1995; Bierwert 1996; Hess 1998). Without these resources this work would not be possible.

2.1 Morphology

Lushootseed (formerly known as Puget Salish) is a Central Coast Salish language whose traditional area ranges from Puget Sound westward to the Cascades. It is a language with a rich morphology, a property it shares with other Salish languages. Largely considered a polysynthetic language, its morphemes are for the most part distinctive, with almost no overlap in function (such as, for example the English *+s* which is a subject marker for verbs, a pluralizer for nouns, and a genetive marker).

In spite of its overall morphological richness, the language's basic system of roots is relatively simple; almost all roots are monosyllabic or disyllabic. Derivational morphology is very common, and transderivationality is pervasive in the language. Most roots can take any inflection (e.g. past tense can occur on nouns and adjectives).

Affixation takes several forms. Prefixation is primarily used for aspect, tense, nominalization, and modality. Suffixation is used to signal transitivity, causativity, voice (active/passive/middle), some aspect, reflexivity/reciprocity, and clause-related subject/object marking. Most of the inflectional morphology involves suffixation. Compounding is not common, though some compounds with relatively high frequency do exist. Incorporation is quite frequent.

Lushootseed is rich in reduplication, which is almost exclusively partial reduplication. It is possible on any part-of-speech category, and there are at least seven different basic types of reduplication: diminutive, distributive, out-of-control, counting people, particularizing, augmentative, and collective.

Lushootseed also has lexical suffixes, a special class of affix that is present in most Salish languages. Lexical suffixes are bound morphemes that have lexical referents. Most commonly they refer to nominal concepts (body parts, geographical entities, common artifacts in daily life, etc.) Around 100 lexical suffixes have been identified for Lushootseed.

Figure 1 gives several Lushootseed words that all derive from the same root morpheme. They involve various processes including prefixation, suffixation, and reduplication.

2.2 Other linguistic aspects

Though this paper focuses on morphology, aspects of other areas of linguistics are important to the topic discussed. A few of these relevant facts are mentioned next.

The language has various dialects including Skagit (and Nooksack), Snohomish, Sauk-Suiattle, and Skykomish (the northern dialects), and Snoqualmie, Suquamish, Duwamish, Muckleshoot (the southern dialects).

Lushootseed phonetics involves a rich consonantal system but relatively few vowels; the entire vocalic inventory involves: a (low), ə (mid), and i, u (high). Consonants include: glides and glottals (w, ẇ, y, ẏ, h, ʔ), plain stops (p, t, k, q) with associated variants involving voicing (b, d, g), glottalization (ṗ, ṫ, ḳ, q̇), and labialization (k^w, $\dot{k}^w$, g^w, q^w, $\dot{q}^w$), affricates (c, ċ, č, $\dot{č}$, d^z, $\check{j}$, $\dot{\check{\lambda}}$), unvoiced fricatives (s, š, x^w, x̌, $\check{x}^w$, ɬ), and lateral resonants (l, l̇).

The orthography involves a (rough) 1-to-1 sound/symbol correspondence, and was

gʷəd	gwEd	*seated*
gʷədil	gwEdil	*sit down*
gʷədtilxʷ	gwEdiltxW	*seat someone, marry*
gʷədis	gwEdis	*sit next to someone*
səxʷgʷədil	sExWgWEdil	chair
səxʷgʷigʷədil	sExWgWigWEdil	*little chair*
gʷigʷədil	gWigWEdil	*sit briefly*
sgʷigʷədil	sgWigWEdil	*brief sitting*
sgʷigʷəilal?txʷ	sgWigWEdilal?txW	*outhouse*
gʷədgʷədil	gWEdgwEdil	*sitting around*
gʷaadil	gWaadil	*people sitting around*

Figure 1: Example of morphological complexity showing various forms derived from the same root. The first column shows the words in traditional orthography, the middle column shows ASCII equivalents used by the program, and the last column shows English glosses.

standardized by Thom Hess in the 1960's. It is largely based on the International Phonetic Alphabet. Usually no upper-case letters are used; instead, proper nouns have underlining (e.g. mali for *(Mary)*, dxʷliləp for *(Tulalip)*, and sqʷali? for *(Nisqually)*). Sounds which are not always pronounced and therefore optional in the orthography are often parenthesized (e.g. t(i) adsgʷa? for *(yours)*). In the work reported in this paper, use of some of these alphabetic characters is not possible or practical, so a close transcription system is used wherein the words are rewritten in the ASCII character set, which involves punctuation characters as well as uppercase and lowercase letters from A through Z.

Various phonological processes are active in the language, including assimilation as well as vowel deletion, lengthening, and reduction. Vowel length is occasionally distinctive (e.g. sdukʷ *(strange thing)* vs. sduukʷ *(knife)*). Stress is fairly regular and thus is usually ignored in orthography. In terms of metrical structure the feet are usual binary, and the meter is most often trochaic.

As regards the lexicon and lexical categories, Lushootseed has few prepositions and adverbs; predicates tend to carry this content. Some function words play several roles: determiners are often used as pronouns, for example.

Many loanwords have entered the language from English, French, Chinook Jargon, and surrounding Native American languages. On the other hand, a rich degree of lexical innovation exists in the language due to the high level of semantic, syntactic, morphological and phonological processes available.

Though the language also has other interesting syntactic, semantic, and pragmatic properties, they will not be addressed in this paper.

3 Computational morphology

The field of computational morphology involves processing morphological structure via computer. This involves parsing (or breaking down or decomposing) a word into its constituent morphemes. Another type of processing is generation, or creating an inflected word from the specification of relevant morphemes and their meaning.

Many methods have been traditionally used in computational morphology; these vary according to the morphological complexity of the languages being processed. For example, a language like English is relatively simple morphologically. Accordingly, a traditional (and still used) approach for English is the "cut-and-paste" method: look up a word in the lexicon; if it doesn't exist, remove a possible affix (e.g. a putative suffix *+s* or *+ed*) and look it up again, possibly adding or removing more letters, and repeat this process until the word is found or all possibilities have been tried (Porter 1980). Given the small number of morphological affixes used in English this process, though *ad hoc*, works fairly well for a language like English.

On the other hand, this type of method is wholly inadequate for processing morphologically complicated languages. The computational complexity of breaking apart the word by stripping off possible morphemes and then accessing a lexicon to verify their status is untenable for long words in a language with many possible morphemes. The process becomes even more intractable when morphophonemic variation occurs across (possible or actual) morpheme boundaries.

In order to respond to the demands of morphological complex languages, the two-level model has been proposed (Koskenniemi 1983). This is a system that views each word in the language as having two simultaneous representations (or correspondences): the lexical and the surface. The lexical representation is characterized as an underlying concatenation of morphemes in their neutral forms (generally corresponding to the lemma). The surface form represents the actual orthographic form of the word in the language. Figure 2 gives examples of lexical/surface pairs for several words in various languages.

Applying the two-level model involves describing and resolving the differences between these levels. This is done by specifying rules, in a fairly common linguistic format, that relate the L:S pairs. These rules are then compiled into a representation called a finite-state table which can be used by computers to build a finite-state automaton. This automaton is used to control the system as it progresses letter-by-letter through a word trying to find correspondences between possible L:S pairs. This formal process, called transduction, need not concern us here; we can conceive of the process as a "black box" that takes linguistic descriptions of Lushootseed morphology and uses them to compute possible L:S pairs.

The two-level approach has been applied to a variety of languages, generally morphologically complex ones, such as Finnish (Koskenniemi and Church. 1988), Turkish (Oflazer 1994), and Arabic (Beesley 1997). An application has also been developed for at least one native American language: Aymara (Beesley and Newton. 1989).

4 The system

In this section we first sketch the basic computational system that implements the two-level model; this system is what constitutes the "engine" of the morphological processor. Then a description of how the system was implemented is given.

The basic system used for the Lushootseed morphology engine is PC-Kimmo (Antworth 1990). It was specially developed for two-level processing of language data,

```
L: #sky#    #sky+s#   #dye+ing#   #die+ing#
S:  sky      skies     dye0ing     dy00ing

L: #Sin+ta#   #yom+ta#   #yob+ta#   #Tug+0ta#
S:  Sin0da     yon0da     yon0da     Tu00ida

L: #travaill+er# #katab+at# #katab+ti#
S:  travaill0es   k0t0b00t   k0t0b0t0
```

Figure 2: Sample lexical/surface correspondence pairs. The top line shows sample English words, the middle line some Japanese forms, and the bottom line shows one French and two Arabic forms. Pound signs in the lexical form represent word boundaries, and plus and minus signs represent morpheme boundaries. Zeros are eventually removed but are included here to illustrate deletion explicitly.

particularly for fieldwork and text analysis tools development. Freely distributed by the Summer Institute for Linguistics[1], it provides a basic level of processing extended by language-specific morphological knowledge sources which are specially developed by the user. PC-Kimmo has been used in processing the morphology of a wide variety of languages via the two-level model.

There are two basic components to the system: (i) the engine itself, which is coded in the C programming language and which the user does not need to delve into; (ii) the knowledge sources including lexicons, rule files, and grammar files. This paper focuses on the latter component, which involves the morphological knowledge sources.

The engine is is capable of two basic modes of operation: (i) recognition, in which a fully inflected word is processed by the system to arrive at a description of the word's morphological decomposition(s); and (ii) generation, in which a specification of underlying morphemes is processed by the system to produce the corresponding surface form(s) of the word.

More information on how to download, install, and run the system is available from the website referred to earlier. A discussion of the principal linguistic knowledge sources (lexicons, rules, and word-formation constraints) for Lushootseed, and their development, follows.

4.1 Lexicon architecture

The system uses a collection of one or more lexicons to represent the basic morphemic inventory of a language. As a word is processed letter-by-letter, the lexicon subsystem is used as a basic device to control and license search through possible sequences of letters and morphemes for a word.

In many implementations, developers use a separate lexicon for each of the possible positions where inflection can take place. When this is done, the user can specify valid sequences of lexicons that should be traversed in order to piece together a word.

[1] see the website at `www.sil.org/pckimmo/`

For the Lushootseed implementation, over ten separate lexicons are used. All Lushootseed data in each lexicon is represented in the ASCII transcription mentioned earler. Each entry in a lexicon consists of the following items of information for a morpheme:

- the lexical form of the morpheme: the underlying representation of the morpheme
- the name of the lexicon
- possible continuation classes for subsequent morphemes: what type(s) of morpheme can linearly follow the morpheme in question
- the gloss of the morpheme: its English translation or gloss
- features that describe, constrain, or pertain to the morpheme in question

Figure 3 shows three sample dictionary entries; on the left, an entry (from the ROOT lexicon) whose lexical form is bəlx̌^{w}, corresponding the English *go by, pass*, which can optionally be followed by of the reduplication patterns; in the middle, an entry from the PROGRSTAT lexicon (which contains forms for progressive and stative inflection) whose lexical form is ləs+, optionally followed by another type of reduplication pattern, and glossed as ProgStatv+; and in the rightmost column, an entry from the VSUFRFX lexicon (which contains forms for reflexive and reciprocal inflection) whose lexical form is +cut, which can be followed by an inchoative marker, and which is glossed +Rfx.

Once the development of lexicons has been completed, the system can process words by analyzing them in terms of simple concatenation of lexicon contents. Thus for example, the system will take a word and match its beginning with one or more morpheme entries from the first possible lexicon. It then progresses across the word, matching the remainder of the word with other possible lexicons and their entries until the word's letters are exhausted. The system's design thus allows for threading through the lexicons in order to find the proper morphemic decomposition of the word. Though useful for many words, this simplistic approach is not adequate, however, so rules are also used by the system.

4.2 Rules

Usually in a given language's morphology, and this is certainly the case for Lushootseed, word formation does not involve simply a direct concatenation of morphemes. Instead, complex interactions may occur at morpheme boundaries. These interactions often involve phonological change, and hence are referred to as

```
\lf ^bElXW                \lf lEs+                  \lf +cut
\lx ROOT                  \lx PROGRSTAT             \lx  VSUFRFX
\alt GetPostRedup         \alt GetPreRedup          \alt GetVSufInch
\gll go by, pass          \gll ProgStatv+           \gll +Rfx
```

Figure 3: Three sample lexicon entries: one for a root (left), one for a prefix (middle), and one for a suffix (right).

morphophonological changes. Lushootseed has a rich inventory of such changes, and accordingly the lexicon subsystem itself is inadequate as a source of knowledge for the engine to use.

For this purpose, a two-level rule capability is integrated into the system. This allows the user to specify any changes that might happen to the basic lexical form of a word and its constituent morphemes, often as a result of phonological or morphological processes. For example, rules can be used to handle allomorphic variants and to specify the changes that happen at the morphophonological interface.

The format for these rules is a close approximation to the rule format used in many phonological textbooks; it involves specifying some kind of change, a delimiter, and then a specification of where the change takes place. The basic format, therefore, is:

change delimiter environment

The delimiter is represented by a type of arrow, which specifies what type of rule is involved: a mandatory one, an optional one, and so forth.

For example, consider a rule in Lushootseed that optionally deletes u at the boundary of two morphemes, as in ɬ(u)adsbəda? (where the optional element is specified with parentheses). The rule to allow both forms (i.e. with and without the optional letter "u") is:

```
RULE
"u:0 => [L|T'] ____ +:@ VW"
```

where the change is the L:S pair u:0 (meaning u deletes), the right arrow signifies that the rule is optional, and the context specifies that the change can take place after the letters ɬ or x̌ and before a morpheme boundary and a vowel.

As stated earlier, these rules are compiled up into finite-state tables and finally a finite-state automaton which the system steps through when analyzing (or generating a word). As the system considers each possible L:S pair, it checks the rule base to see if any might apply to the pair in question, and applies them appropriately. If the system enters the automaton and is unable to reach an end state when all possibilities are exhausted, processing fails; otherwise, success is attained and a morphological parse (or generated surface form) is created. Many rules are usually needed to describe these changes for any language, and these rules operate in parallel.

A special file called the rule file contains the rules and associated information used by the system. First of all, the rule file declares the alphabet (i.e. the characters) that will be used in the word forms. This is where the ASCII transcription scheme mentioned earlier is defined.

In addition, the rule file contains a declaration of the special symbols used in rule formulation. The Lushootseed engine uses the standard special symbols: 0 (zero) for a null character (i.e. one that is not used in orthography), * (wildcard) for any character, and # for the word boundary.

The rule file also supports declaration of special subsets that might be useful in rule specification in cases where several characters or sounds group together. For example, Lushootseed uses (among others) VW for the set of vowels, SIB for the sibilants, BILAB for bilabials, and CONS for consonants.

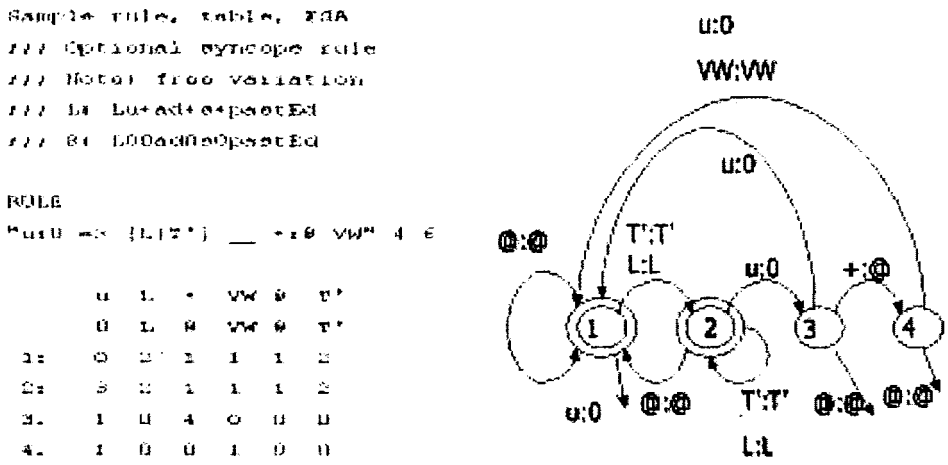

Figure 4: A sample rule for optional syncope in Lushootseed. On the left are: comments including a typical relevant lexical/surface pair, a specification of the rule itself, and a 4-by-6 finite-state transition table corresponding to the rule. On the right is the automaton implementing the rule and state table.

Most importantly, though, the rules file also contains the rules declarations themselves. A few dozen rules have been implemented for Lushootseed. Figure 4 shows the rule discussed earlier, its corresponding finite-state table, and its finite-state automaton.

4.3 Word grammar

Another knowledge source used by the system is a word-formation grammar. This is a context-free grammar used to specify word-level constraints on order, structure, cooccurence of morpheme classes. It imposes a further level of constraint to word structure beyond that supplied by the lexicon architecture and rule base. The grammar rules follow standard conventions for phrase-structure implementations: a left-hand category precedes an arrow which precedes one or more elements to be combined to form the left-hand constituent. Constraints are enforced by unification-based comparison of compatible features. These features originate in the lexicon and can percolate through the word-formation process to control morphemic composition at any stage.

Following is a sample word-structure rule, which allows a noun to be formed from a verb frame, root, noun, or verb to which an enclitic determiner is added.

```
RULE
NWord -> { VFrame / VWord / NWord / RootX } DET2
```

Another byproduct of the word grammar is the ability to specify and produce a tree-based, graphical representation of the word's constituent structure. In addition, various levels of glosses can be specified.

5 Current status and sample outputs

At the present time almost all of the morphemes of the language have been entered into the system and can be processed successfully. This includes all lexical suffixes, all affixes, and a majority of the root forms listed in the canonical dictionaries for Lushootseed. Additionally, the lexicon architecture has been put in place. Consequently, all words that involve straightforward morpheme concatenation are successfully handled by the system.

In addition, several morphophonemic alternations are handled via the few dozen rules that have been added to the system. Some of these rules, particularly the reduplication rules, are rather complex. All of the major alternation patterns have been implemented (e.g. epenthesis, assimilation, reduction, deletion, reduplication, etc.), though specific patterns remain to be done. Three of the six reduplication patterns have been implemented.

Consider, for example, the word gʷədsəutudᶻildubut. When presented to the system with the command "recognize", it will take the word and calculate its morphemic decomposition (shown on the left), and its morphemic gloss (shown on the right):

```
PC-KIMMO>recognize gWEdsutudZildubut
gWE+d+s+?u+^tudZil+du+b+ut    Dub+my+Nomz+Perf+bend_over+OOC+Midd+Rfx
```

Similarly for the word adsukʷaxʷdubs:

```
PC-KIMMO>recognize adsukWaxWdubs
ad+s+?u+^kWaxW+du+b+s      Your+Nomz+Perf+help+OOC+Midd+his/hers
```

Generation is accomplished by specifying on the input a set of morphemes in their lexical representation and using the command "generate"; the system takes the input and produces one or more surface forms for the word. Sometimes this is straightforward, as for the word adpastədal?txʷ, which involves a prefix, the root, and a lexical suffix:

```
PC-KIMMO>generate ad+^pastEd=al?txW
adpastEdal?txW
```

Other times, the process can become more complicated, as with the word adsukʷaxʷdubs, which drops the glottal stop:

```
PC-KIMMO>generate ad+s+?u+^kWaxW+du+b+s
adsukWaxWdubs
```

As mentioned earlier, word-structure diagrams can also be produced via the word grammar component. Currently the grammar has over 30 rules which constrain morpheme cooccurrence and describe basic constituency stages. Figure 5 shows a sample word-structure parse tree produced by the system for the Lushootseed pseudo-word ɬubələskʷaxʷyildutəxʷčəɬ.

```
PC-KIMMO>recognize LubElEskWaxWyildutExWCEL
Lu+bE+lEs+^kWaxW+yi+il+d+ut+ExW+CEL
Fut+ANEW+PrgSttv+help+YI+il+Trx+Rfx+Inc+our

1:
```

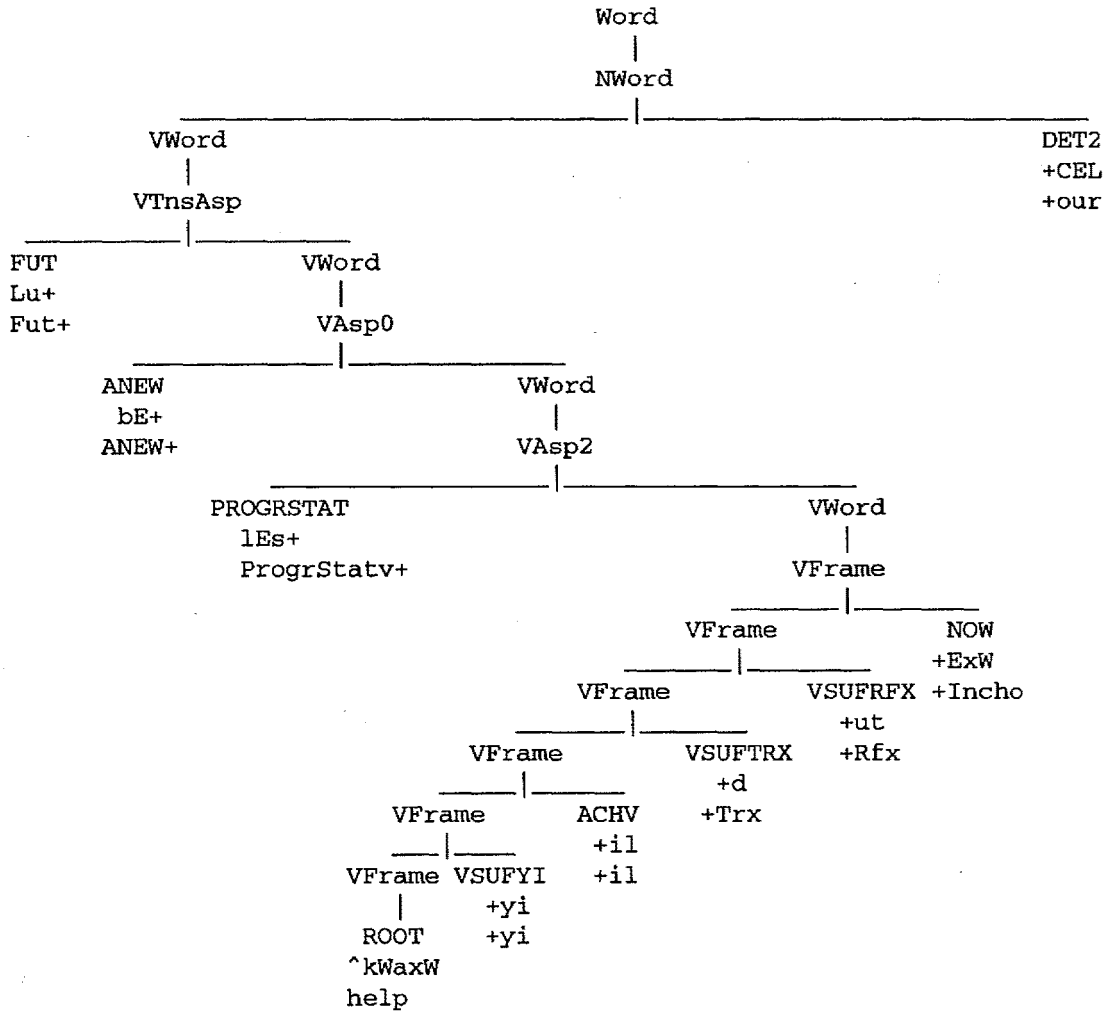

Figure 5: A word-structure parse tree for an inflected pseudo-word in Lushootseed. The word is improbable, but shows how a complex concatenation of morphemes can be analyzed and diagrammed.

6 Future work and possible applications

Though the core functionality of the engine has been implemented, there are still items that require work before the system can be considered completed.

For example, some lexical morpheme entries still need to be added to the lexicon. This is particularly true for dialectal variants, which to this point have not been addressed. In addition, English glosses need to be integrated with most of the entries. Testing coverage still needs to be done in a more methodical manner than done so far.

While all commonly occurring morpheme sequences are currently recognized by the system, there are several words that do not seem to conform to the standard patterns discussed in the literature. This is particularly true for reduplication patterns. For these problematic forms more knowledge source development has to be done. In addition, the system does not yet handle incorporation.

The two-level system is well suited for computing other phonologically-related properties of a language than those involving morphology; suprasegmental phenomena such as stress (and related alternations) could also be addressed, but so far have not been.

A morphology engine like the one described here can be used in a variety of settings. As mentioned earlier, it was designed originally to serve as a tool for fieldwork and for corpus and lexicon management. Morphological engines have also been deployed in language-learning settings within tutoring programs, testing software, and language-game entertainment software. Perhaps this engine might be helpful in similar settings for future language research or computer-assisted Lushootseed language instruction.

Morphological processing is a relatively low-level but crucial step in several types of text-based natural language processing. For example, many search engines and corpus manipulation tools include a morphological processing component. It is expected that this engine will be useful in corpus-based analysis of Lushootseed wordforms, inflection, derivation, and lexical usage. Almost all syntactic parsers also assume prior morphological processing or integrate it into the parsing process. This engine produces feature structures which are compatible with several widely available syntactic parsers, and thus the construction of full-text parsers (and also generators) for Lushootseed becomes more feasible.

Speech applications also rely on morphological engines, for both speech recognition (where morphophonological clues are crucial for constraining the search space of processing possibilities), and also for speech synthesis (where critical suprasegmental properties are often determined from morphological structure). A morphology engine should enable principled development of future speech-based Lushootseed applications.

Finally, the approach used for the Lushootseed engine should serve well in the development of similar processors for handling other Salish languages.

References

Antworth, E. 1990. *PC-KIMMO: a two-level proecssor for morphological analysis.* Number 16 in Occasional Publications in Academic Computing. Dallas, TX: Summer Institute of Linguistics.

Bates, D., T. Hess, and V. Hilbert. 1994. *Lushootseed Dictionary*. University of Washington Press.

Beesley, K. 1997. Finite-state descriptions of Arabic morphology. In *Proceedings of the Second Cambridge Conference: Bilingual Computing in Arabic and English*, Literary and Linguistic Computing Center, Cambridge University, UK.

Beesley, K. and S. Newton. 1989. Computer analysis of Aymara morphology. In *Proceedings of the 15th Annual Deseret Language and Linguistics Symposium*, pp. 126–144. BYU.

Bierwert, C. 1996. *Lushootseed texts: An introduction to Puget Salish Narrative Aesthetics*. Studies in the Anthropology of North American Indians. University of Nebraska Press.

Hess, T. 1976. *Dictionary of Puget Salish*. University of Washington Press.

Hess, T. 1995. *Lushootseed reader with introductory grammar, Vol. 1*. University of Montana Occasional Papers in Linguistics. Summer Institute of Linguistics.

Hess, T. 1998. *Lushootseed reader with introductory grammar, Vol. 2*. University of Montana Occasional Papers in Linguistics. Summer Institute of Linguistics.

Hess, T. and V. Hilbert. 1977. *Lushootseed 1 and 2: The language of the Skagit, Nisqually, and other tribes of Puget Sound*. Daybreak Star Press.

Koskenniemi, K. 1983. Two-level model for morphological analysis. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, pp. 683–685.

Koskenniemi, K. and K. Church. 1988. Complexity, two-level morphology and Finnish. In *Proceedings of the 12th International Conference on Computational Linguistics*, pp. 335–340. Association for Computational Linguistics.

Oflazer, K. 1994. Two-level description of Turkish morphology. *Literary and Linguistic Computing 9*(2).

Porter, M. 1980. An algorithm for suffix stripping. *Program 14*(3), 130–137.