Building Search Engines for Algonquian languages[1]

MARIE-ODILE JUNKER AND TERRY STEWART
Carleton University

Résumé

L'élaboration de ressources linguistiques qui aident à la préservation des langues amérindiennes offrent bien des défis. Bien qu'on compile des dictionnaires, les locuteurs et locutrices sont souvent bien en peine de les utiliser. La standardisation récente de l'orthographe, la prédominance de la langue orale et une grande variation dialectale font que trop souvent, les gens ne trouvent pas les mots qu'ils cherchent car ils les écrivent de travers ou juste différemment. Nous montrons ici comment nous avons construit un moteur de recherche pour le dictionnaire cri de la Baie James sur le web (www.eastcree.org) qui permet les fautes d'orthographe ou les orthographes créatives, et comment nous l'avons ensuite incorporé dans un moteur plus complexe pour la recherche de verbes à partir de formes verbales fléchies Nous montrons aussi comment nous avons adapté ces outils à une langue voisine, l'innu (ou le montagnais). Notre solution est de combiner deux approches computationnelles à la correction de l'orthographe (en mesurant la différence entre le mot entré et les mots du dictionnaire, et en appareillant la phonétique), et de les adapter aux langues algonquiennes à partir de connaissances linguistiques. Ce moteur pourrait servir de modèle et être adapté pour d' autres langues algonquiennes ou minoritaires.

INTRODUCTION

Building effective resources that help preserve Native American languages presents many challenges. Although resources like dictionaries are being compiled, speakers often are at a loss in using them. The recent standardization of orthography, the predominance of an oral language, and considerable dialectal variation mean that speakers often do not find the words they are looking for because of incorrect or non-standard spelling of words. We show here how we built a search engine for the Eastern James Bay Cree dictionary on the web (www.eastcree.org) that allows

Papers of the 39th Algonquian Conference, eds. Karl S. Hele & Regna Darnell (London: The University of Western Ontario, 2008), pp. 378-411.

spelling mistakes or creative spelling, and how we further incorporated it into a more complex engine for verb dictionaries, where an inflected form has to be matched with a basic dictionary form. We also show how we then adapted our algorithm to the Innu language.

East Cree is an Aboriginal language spoken in the Eastern James Bay region, in Quebec, Canada. There are 13,000 speakers, living in 9 communities spread over a very large territory. Internet resources are increasingly used to bridge physical distances between communities. The East Cree language is taught in the schools under the administration of the Cree School Board. The first version of the East Cree dictionary (Cree-English only) was published in 1987, and the latest version is available on the web at http://dict.eastcree.org. The web dictionary publication was made possible by a participatory action research project, called eastcree.org, which explores how information technology can help language documentation and preservation. The web version has allowed the editorial team to make regular updates since its first release in 2004. A Cree-French version was released in 2007 and is now also being updated regularly.

There are two dialects of East Cree, the Northern and the Southern dialect, the latter being further differentiated between a Coastal variant and an Inland variant. East Cree is part of a linguistic continuum ranging from Innu, Montagnais, and Naskapi in the East, and all the Cree dialects from James Bay to the prairies in the West.

Cree has a rich thousand-year oral tradition. Although syllabic orthography (see appendix A) was established around 1860, and used sporadically for a few religious texts and private correspondence, the written language came into its own around 1970. The standardization of the orthography is an on-going process, with speakers of different ages or communities using different forms of spelling. The orthography used in the Cree dictionary is the one developed for the Cree School Board by

MacKenzie et al. (2004) for the Southern dialect and Salt et al. (2006) for the Northern dialect[2]. Cree dictionary entries and examples are given in both syllabic and roman orthographies. Systematic conversions from roman to syllabic orthographies are implemented in the web dictionary.

The web dictionary includes the two dialects (Northern and Southern) and allows searches in Cree syllabics, Cree roman, English and French. An electronic dictionary of inflected verb forms is in preparation and will soon complement the existing Web Dictionary for both dialects.

The remainder of this paper is organized as follows: First we discuss the kind of spelling difficulties encountered by Cree speakers. Second, we explain our solution and how it works. Then, we describe the computational aspects of the relaxed search and the verb inflection search. Next, we show how we adapted these tools to the Innu language. Lastly, we briefly discuss some usage statistics on how such dictionaries are used.

THE PROBLEMS

When using a dictionary, speakers often do not find the words they are looking for. Users of the Cree-English book dictionary reported not being able to find some Cree words. They may of course be looking for a word that is just missing from the dictionary, or they may be searching a word using one of its inflected forms. The most common reason, however, is poor or alternative spelling. Evidence for spelling difficulties was gathered by the first author by observing fluent speakers, from second-language students (herself included), from reports by Cree literacy instructors, and from explicit instructions found in the Spelling Manuals by Salt et al. (2006) and MacKenzie et al. (2004). There is a number of common spelling problems we needed to address. We discuss them one by one, using relevant examples, many of them coming from the Spelling Manuals. We then turn to the problems of inflected forms.

Vowel length

There are seven vowels in Southern East Cree, four long vowels and three short ones, and this is reflected in the spelling. Northern East Cree does not have the vowel *e*. Long vowels are indicated by a dot over the syllabic symbol and by hats (usually in print), or by double vowels (usually on the web) in the roman orthography. The vowel *e* does not have a short counterpart.

A common spelling problem arises when users fail to indicate the proper vowel length. For example, they will search for ᓇᐯᐤ *n**a**peu* instead of ᓈᐯᐤ *n**aa**peu*. Since *napeu* does not exist, they will not find it. Or they will search for one of: ᐊᑎᑯ ***a**chikw* or ᐋ̇"ᑎᑯ ***aa**hchikw* when both exist.

Sometimes the vowel length problem can come from dialectal differences: In the Northern dialect the following word has a long vowel: ᓱᓂᒋᐧᓂᐦ *saani**tii**paanish*, while in the Southern dialect it has a short vowel: ᓱᓂᒍᐧᓂᐦ *saani**ti**paanish.*

Vowel quality

Another common problem comes from dialectal differences in vowel quality. As shown in the following examples, an *e* in a Southern dialect is pronounced and written *aa* in the Northern dialect. An *a* in the Southern dialect corresponds to an *i* in the Northern. Since many speakers move from one community to another today, some confusion occurs.

Southern East Cree		**Northern East Cree**	
ᑲᐸᑦ	k**a**pat	ᑭᐱᑦ	k**i**pit
ᒪᔑᓇ"ᐃᑲᓵ	m**a**sin**a**hiikan	ᒥᓯᓂ**i**hiik**i**n
ᓇᒣᐦ	n**a**m**e**s	ᓂᒪ**aa**s
ᔕ·ᐁᔭᒣᐤ	sh**a**weyim**e**u	ᔑ·ᐊᔭᒪᐤ	sh**i**w**aa**yim**aa**u

The vowel quality problem is also apparent in the spelling of finals. As recommended in the 2004 (Southern dialect) Spelling Manual:

For most words ending in ᐦ be careful to write ***ᐱ, ᑎ, ᓱ, ᓯ pi, ti, shi, ni*** *and not* ***ᐳ, ᐢ, ᓴ, ᓐ pu, tu, shu, nu*** *before final* ***ᐦ kw****.*

Write…		***Not…***	
ᒥᔨᑎᐣ	mish**t**ikw	ᒥᔨᑐᐣ	mish**t**ukw
ᑭᓐᐱᐣ	chinep**i**kw	ᑭᓐᐳᐣ	chinep**u**kw
ᑮᔮᐣ	chiish**i**kw	ᑮᔭᐣ	chiish**u**kw
ᐅᔨᑮᔮᐣ	ushchiish**i**kw	ᐅᔨᑮᔭᐣ	ushchiish**u**kw

Voice/voiceless consonant differences (not phonemic in Cree)

The voice/voiceless consonant difference is not phonemic in Cree, thus it is not reflected in the orthography. However, biliteracy with English or French where this difference is represented, leads to its representation in the Cree roman orthography. Voiced consonants are thus sometimes introduced in the roman orthography. The following consonants are concerned: p/b, t/d, k/g, s/z, (sh) [ʃ / ʒ], (ch) [ʧ /ʤ], f/v.

For example, someone will search in Cree roman, typing *naa**b**eu* instead of *naa**p**eu*. There is actually quite a tradition of representing this non-phonemic difference in the roman spelling of place names in Quebec: *Waska**g**anish* instead of Cree *Waska**k**anish*, *Oujé-Bou**g**oumou*, *Chisasi**b**i*, *Wahpma**g**oostui*.

Truncating words

Some words are truncated. Sometimes people forget dots in front of syllabic symbols, or a *w* before a vowel in roman. *Dots are written before a syllabic symbol and indicate a w sound before the vowel, or between the consonant and the following vowel.* (Spelling Manual, 2004)

Words <u>without</u> the dot or *w*		**Words <u>with</u> the dot or *w***	
ᐊᑭᑲᐣ	**a**chikaash	ᐧᐊᑭᐣ	**wa**chishkw
ᐃᔦᔨᑯᐅ	**i**yeskupuu	ᐧᐃᔦᐣ	**wi**yesh
ᐁᐣ	**e**shkw	ᐧᐁᐣᑲᐠ	**we**shkach
ᓰᐠ	**chaa**k	ᐧᓰᑲᓪᒡ	**chwaa**kalit
ᒣᐧᑲᐠ	**me**kwaach	ᐧᒣᐦᐠ	**mwe**hch
ᑮᓀᐤ	chii**naa**u	ᑭᐧᓀᐤ	chi**nwaa**u

Finals are small syllabic symbols at the end of the word and tend to be forgotten:

ᒥᑎᐦᒋᐦ	mitihchii**h**
ᐊᑯᐦᐸ	aku**hp**
ᐊᒥᕽᑯ	ami**skw**
ᒥᓱᐦᑯ	misaa**hkw**

Or sometimes people mix *k* and *kw* at the end of a word:

Final *k*		Final *kw*	
ᒪᐢ	maa**k**	·ᒪᑯ	mwaa**kw**
ᐃᓴᑯᑕᐢ	ishkuta**k**	ᐃᓴᐱᒥᐦᑎᑯ	ishpimihti**kw**
ᓀᐢ	che**k**	ᓀᑯ	che**kw**
ᑯᐯᐢ	kupe**k**	·ᐃᓂᐯᑯ	wiinipe**kw**

Another difficulty is to not forget a little *h* in the middle of a word. Sometimes there are two distinct words, one with *h* and one without:

With " (h)		**Without " (h)**	
ᒥᐦᑯ	mi**h**kw	ᒥᑯ	mikw
ᓯᐦᐠ	sii**h**tuu	ᓯᐠ	siituu
·ᐃᐦᐸᐤ	wii**h**paau	·ᐃᐸᐤ	wiipaau
ᐅᐦᓀᐤ	uu**h**cheu	ᐅᓀᐤ	uucheu
ᔭᐦᓯᐤ	yaa**h**yeu	ᔭᓯᐤ	yaayeu
ᒪᐦᑯᖨᐤ	ma**h**kusheu	ᒪᑯᖨᐤ	makusheu
ᒪᐦᐹ	maa**h**kii	ᒪᐹ	maakii
ᐯᐦᒑᐤ	pe**h**taau	ᐯᒑᐤ	petaau
ᐅᐦᐢᐱᒪᒡ	u**h**chipitam	ᐅᐢᐱᒪᒡ	uchipitam

The final *u* poses two challenges. Sometimes it is written, sometimes it is not. So there are two possible mistakes: forgetting or adding the final *u. There are some sounds for which no final exists. In this case a large syllabic without a dot has been used, but is not fully pronounced, much like the final* ᑯ. (Spelling Manual, 2004)

Large syllabic for the final *u*	
ᓂᐦᔨ	niish**u**
ᓂᑎᑐ	nisht**u**
ᓂᐃᔭᐤ	niyaay**u**

In a few number words a final 'u' sound may be heard, but there has been a decision not to write it. (Spelling Manual, 2004)

Final *u* not written	
ᒥᑦᑕᐧᐦᑕ	mitaaht
ᐯᔭᑯᓱᐢᐠ	peyakushaap

The initial syllable of a word is also a spelling difficulty since it is sometimes an unpronounced syllable:

You hear and want to write...		Write...	
ᑦᐤ	taau	ᐃᐧᑦᐤ	**ih**taau
ᑕᑯᓇ	takun	ᐃᐧᑕᑯᓇ	**ih**takun
ᑐᑕᒪ	tuutam	ᐃᐧᑐᑕᒪ	**ih**tuutam
ᓭᑐᑎᓇ	shtutin	ᐊᓭᑐᑎᓇ	**a**shtutin
·ᖿᐤ	kweu	ᐃᓴ·ᖿᐤ	**i**skweu

Because of pronunciation, people often shorten the ᓂ *ni* at the beginning of a word written fully.

Write…		Not…	
ᓂᐱᐧᑯᔨᓇ	**ni**tuhkuyin	ᓇᐱᐧᑯᔨᓇ	ntuhkuyin
ᓂᑭᑯ	**ni**chikw	ᓇᑭᑯ	nchikw
ᓂᐁᐧᐅᐁ	**ni**tuuhuu	ᓇᐁᐧᐅᐁ	ntuuhuu

Some of the number words are pronounced two ways, with or without ᓂ *ni* at the beginning.

You hear and want to write...		You also hear and write...	
ᑯ·ᑫᓂᓴ	kutwaashch	ᓂᑯ·ᑫᓂᓴ	**ni**kutwaashch
ᔭᓇᓇᓄᐤ	yaanaaneu	ᓂᔭᓇᓇᓄᐤ	**ni**yaanaaneu

Older way of spelling words, still used

Because the Cree orthography has changed, some people still remember and continue to use the old way of spelling words. Common problems are:

- the old roman orthography *o* replaced by syllabic ᐧᐊᑊ° *waau*.
- uncertainty about *i* and *yi*
- old *yuu* as verb ending, instead of modern indication of verb stem morphology *yiu* (Northern dialect)

Examples:

o/waau:	nisht**o**	>	ᓂᔨᐧᑦ**waau**	nisht**waau**
ei/eyi:	ᐃᑌᐞᒣ° ite**i**meu	>	ᐃᑌᐳᒣ° ite**yi**meu	ite**yi**meu
ai/ayi:	ᐊᑊᐞᐱᑌ° aa**i**piteu	>	ᐊᑊᐳᐱᑌ° aa**yi**piteu	aa**yi**piteu
yuu/yiu:	ᐱᓯᐱᓴ pisupi**yuu**	>	ᐱᓯᐱᐳ° pisupi**yiu**	pisupi**yiu**

The old orthography often did not include small syllabic symbols. As discussed above, these too might be missing if people use the old spelling.

French spelling of words

Many Cree people today are literate in French. Because the roman orthography for Cree is based on an English spelling of phones, common mistakes include:

- tch for ch [ʧ]
- dj for ch [ʤ]
- j for sh [ʒ]
- ch for sh [ʃ]

Inflected words

A set of different problems arises when considering inflected words. A dictionary cannot contain all inflected forms of all nouns and verbs; instead, it generally will only contain verbs and nouns in their standard citation form. For example, the dictionary citation form for verbs in East Cree is the third person indicative neutral. This means that even if someone correctly spells a fully conjugated verb, avoiding all of the above potential spelling

errors, it will still not be found in the dictionary. We cannot expect all users to know in advance what the dictionary citation form is. Therefore one goal of our search engine was to also work on inflected words.

THE SOLUTION

Our goal was to allow users to type Cree words using possible alternative spellings and still be successful in finding the words for which they were searching. As such, we also wanted users to develop an awareness that their spelling was not standard.

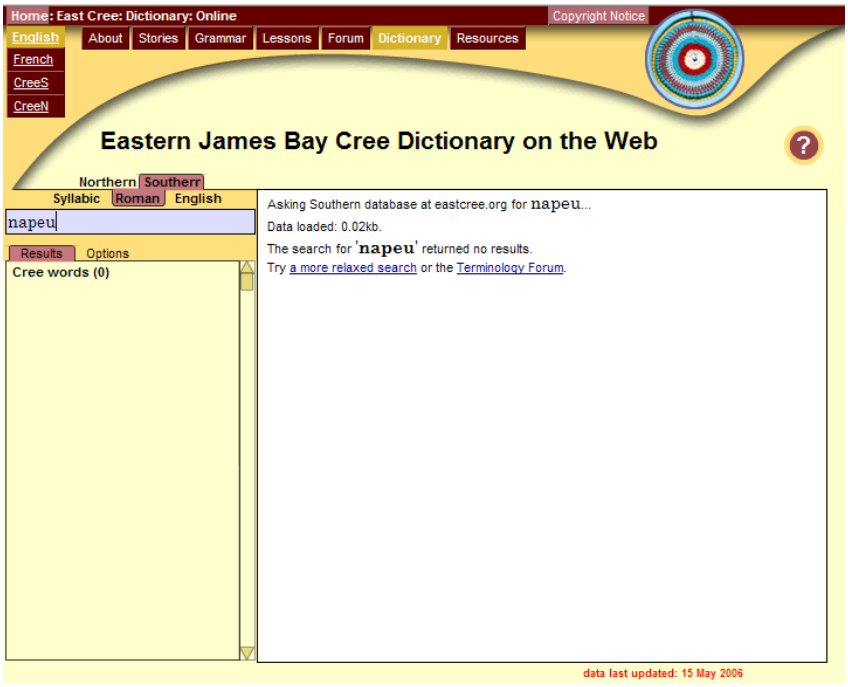
We thus had to build an extended search engine that allowed a different spelling than the one used in the dictionary. At the same time, we had to keep the regular search and make the extended search available only after a regular search was unsuccessful. Another constraint was to limit the number of results to a small number, and allow them to grow incrementally, with user control.

To do this, we could not make use of previously existing spelling software. Although there has been extensive research and development in spelling correction tools for languages such as English, these tools are built around common English misspellings, and so would not give suitable suggestions if applied to Cree. Furthermore, the complex inflections found in the Cree language are not seen in English. Tools developed for other mainstream languages (such as German, which does exhibit complexities such as compound words) are highly specific to that language, and are developed via extensive linguistic analysis as well as examining the error patterns of a large user base. This is not a feasible approach for Algonquian languages.

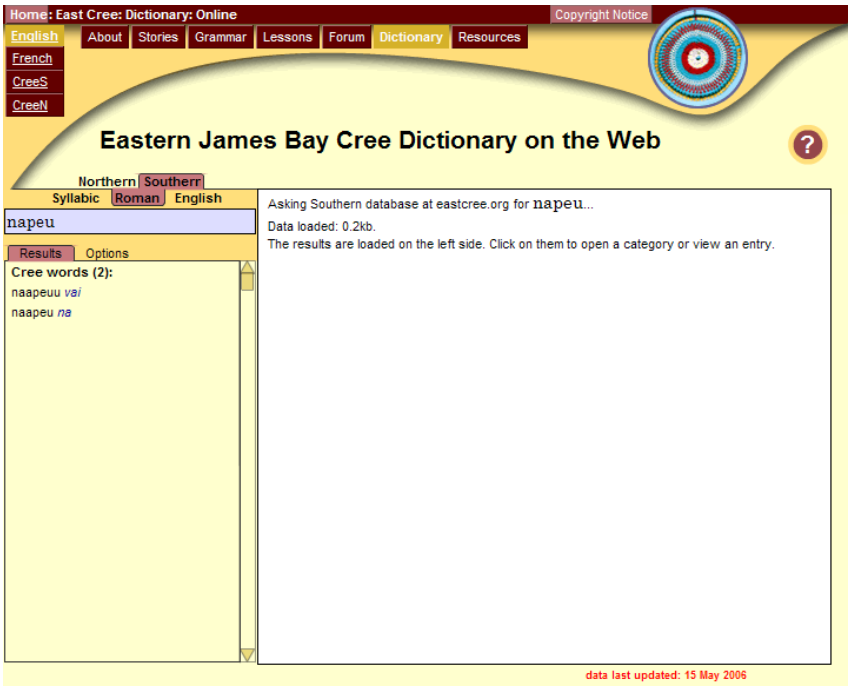
For simplicity, our current system is divided into two parts. First, we have the standard dictionary search system. This is meant to be used much like a normal dictionary, but it is capable of handling common misspellings of words, suggesting words in the dictionary that are “close to” what the user has entered. Second, we have a separate system for identifying inflected words. This

allows, for example, a Cree speaker to type in an inflected verb, and it will indicate the stem and the verb form. This verb search system is also capable of handling common misspellings, and indeed uses the same algorithms as the dictionary search[3].

The following screen shows an example of an unsuccessful search using the standard dictionary. Here, the user has searched for *napeu*, which is an incorrect spelling of *naapeu* 'man', and so no results are found. The system then suggests that they try a "relaxed" search, which incorporates the alternate spellings.



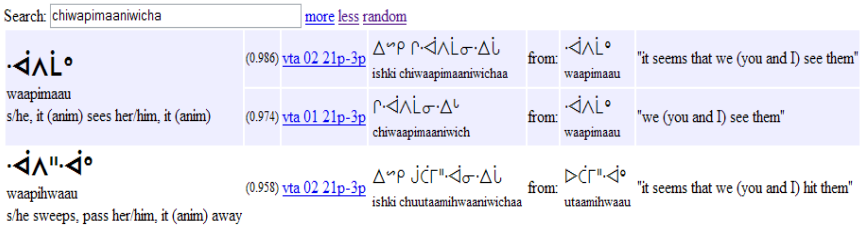
The resulting relaxed search shows two possible options, *naapeu* and *naapeuu*. Importantly, it does not show *napet* 'alas', which is orthographically similar to *napeu*, but does not represent a common spelling error.



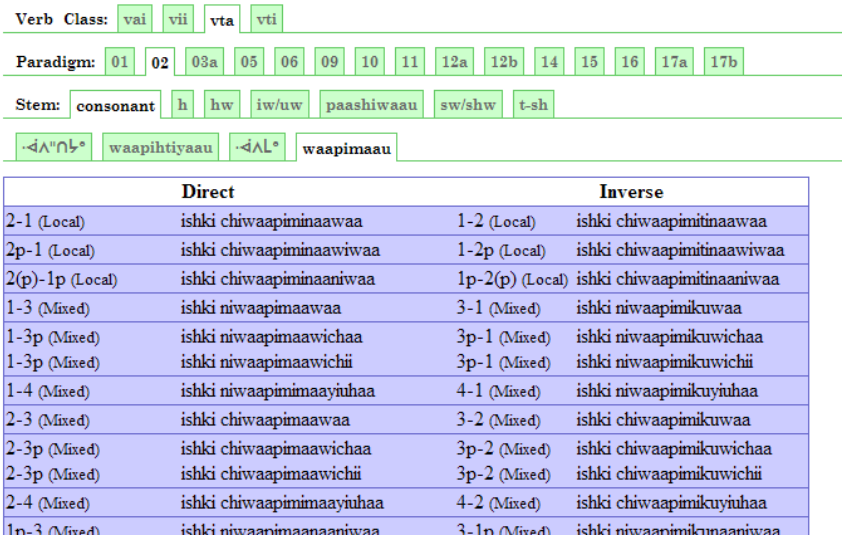
The same search could have been performed in syllabics, with the user omitting a dot on top of the first syllabic character (ᓇᐯ° instead of ᐃᓇᐯ°). Since there is an exact correspondence between the roman and syllabic forms, we are able to convert from one format to the other. The algorithm we have developed uses the roman form, with additional conversions for syllabics.

The separate verb inflection search system follows a similar approach, although its interface is not yet as developed as the one for the dictionary search. An inflected verb is entered into the search box using the roman orthography. The result provides a list of suggestions for the stem of the verb (sorted in order of likelihood, given common spelling patterns). It also lists what form of verb has been provided. In the following example, the incorrectly spelled verb *chiwapimaaniwicha* has been entered. The system has correctly identified the verb stem as *waapimaau* 'to see', but it further noted that it could also be a highly misspelled

version of *waapihwaau* 'to sweep away'. The verb form is identified as VTA in the 21p-3p person, and the best match is to paradigm 02, followed by paradigm 01[4]. It should be noted that the system has correctly identified paradigm 02, even though the user has not provided the *ishki* particle. Beside this information, an example of this verb inflection is given, to aid understanding. This is always to an attested example in our verb paradigm database, which contains an example of each verb stem type in each paradigm.



The verb search system also provides a visual display of the example verbs in the database. This is viewable in either roman or syllabics.



	Direct		Inverse
2-1 (Local)	ishki chiwaapiminaawaa	1-2 (Local)	ishki chiwaapimitinaawaa
2p-1 (Local)	ishki chiwaapiminaawiwaa	1-2p (Local)	ishki chiwaapimitinaawiwaa
2(p)-1p (Local)	ishki chiwaapiminaaniwaa	1p-2(p) (Local)	ishki chiwaapimitinaaniwaa
1-3 (Mixed)	ishki niwaapimaawaa	3-1 (Mixed)	ishki niwaapimikuwaa
1-3p (Mixed)	ishki niwaapimaawichaa	3p-1 (Mixed)	ishki niwaapimikuwichaa
1-3p (Mixed)	ishki niwaapimaawichii	3p-1 (Mixed)	ishki niwaapimikuwichii
1-4 (Mixed)	ishki niwaapimimaayiuhaa	4-1 (Mixed)	ishki niwaapimikuyiuhaa
2-3 (Mixed)	ishki chiwaapimaawaa	3-2 (Mixed)	ishki chiwaapimikuwaa
2-3p (Mixed)	ishki chiwaapimaawichaa	3p-2 (Mixed)	ishki chiwaapimikuwichaa
2-3p (Mixed)	ishki chiwaapimaawichii	3p-2 (Mixed)	ishki chiwaapimikuwichii
2-4 (Mixed)	ishki chiwaapimimaayiuhaa	4-2 (Mixed)	ishki chiwaapimikuyiuhaa
1p-3 (Mixed)	ishki niwaapimaanaaniwaa	3-1p (Mixed)	ishki niwaapimikunaaniwaa

RELAXED DICTIONARY SEARCH

The basis of both the dictionary search and the inflected verb search is what we refer to as a "relaxed" search. This allows us to find a particular word even if it has been misspelled (i.e. it allows for a more "relaxed" approach to spelling, as opposed to a standard dictionary lookup where the word one is looking for must be spelled correctly). To implement this system, we adapted and combined two common computational methods for comparing spellings of words, Levenshtein Distance (the orthographic similarity between words) and Phonetic Matching (indicating that certain phonemes are 'closer' to each other than other phonemes). These are tools common to commercial-grade spell-checking systems, but, as we show, they can be used without the extensive research and development commercial systems require.

Levenshtein distance

The main measure for comparing two sequences of letters (i.e. words) to determine how different they are is based on the simple intuitive measure of counting how many letters are different. For example, *abcde* and *abxye* have three letters in common and two letters that are different. The *c* is changed to an *x*, and the *d* is changed to a *y*. We can thus reasonably say that the distance between these two sequences of letters is 2. This sense of difference is known as the Hamming Distance (Hamming, 1950). However, with this approach it is unclear how to treat the difference between *abcde* and *bcde*. Here, we have deleted one letter, and the naïve Hamming Distance approach would indicate that all the letters are different (the *a* has changed to a *b*, the *b* has changed to a *c*, and so on).

A more complex algorithm for comparing these sequences of letters is the Levenshtein Distance (Levenshtein, 1965). Here, instead of considering only substitutions of letters, we also consider insertions and deletions. The algorithm identifies the minimum number of edits (substitutions, insertions, and deletions)

which will convert the one word to the next. This lets us say that the difference between *abcde* and *bcde* is 1 (a single deletion), and the difference between *pzzel* and *puzzle* is 3 (inserting a *u*, inserting an *e* at the end, and deleting the earlier *e*).

The actual implementation details of the Levenshtein Distance algorithm are beyond the scope of this paper. However, it should be noted that this is a common computer science tool, and most programming languages have freely available, highly optimized versions of the algorithm, thus making it a very fast calculation.

It is possible to use Levenshtein Distance on its own as a spell-checking system, by simply finding all the words in the dictionary that have a difference of 1 or 2 (or some other limit) from the word a user has entered. However, this would be a very strict system which does not make use of the knowledge of common errors in the language. For example, a user typing *napew* will find a closer match to *napet* (at a distance of 1) than to *naapeu* (at a distance of 2). To make use of this sort of knowledge, we need to enhance the system with Phonetic Matching.

Phonetic Matching

While the Levenshtein algorithm is highly efficient at determining the difference between two words, it is completely language and encoding neutral. There is no sense in which a *b* and a *p* are more similar than, for example, a *q* and a *i*. Any difference of any sort is considered to have an equivalent value, whether it be changing a vowel to a consonant or simply changing the length of the vowel. What is needed is a language-dependent system to allow for the specification of the importance of various changes.

One existing system that has received wide usage in North America is *Soundex* (Russell, 1918). Since 1920, this has been used in the U.S. Census to manage the task of finding similarities between people's last names. This was specifically designed so that changes such as *Smith* to *Smyth* would be ignored by the census system. This was accomplished be defining a way of

encoding any last name into a sequence of numbers. This encoding was done with the following rules:

1. Convert these letters to the number 1: *bfpv* (labials)
2. Convert these letters to the number 2: *cgjkqsxz* (gutterals and sibilants)
3. Convert these letters to the number 3: *dt* (dentals)
4. Convert these letters to the number 4: *l* (lateral liquid)
5. Convert these letters to the number 5: *mn* (nasals)
6. Convert these letters to the number 6: *r* (retroflex liquid)
7. Remove any number that is beside an identical number
8. Remove all the following letters: *aehiouwy*

The result is a numerical code representing a particular name. Both *Smyth* and *Smith* would become 253, as would *Smit*, *Smithe*, *Smithy*, *Smythe*, and so on. By organizing the census information by these Soundex codes, all of the data for these related names would be placed in the same location. However, other names such as *Knot* would also have a code of 253. In the official Soundex algorithm, this is partially dealt with by keeping the first letter of the name (which seldom changes), and a similar problem with long names is dealt with by only considering the first 4 digits. This results in the official Soundex code for *Smith* being S530.

Unfortunately, such an approach is perhaps too general for our situations. Soundex leads to exact matches for all of *Powers*, *Pierce*, *Price*, *Perez*, and *Park*. We need a system which is somewhat intermediate between these two approaches. With Levenshtein Distance, any change is equivalent. With Soundex, any vowel change and any consonant change within the groups identified above is completely ignored. A more useful system would allow for the specification of intermediate differences.

An additional problem of the Soundex system is that it was developed for English pronunciation and spelling. Some of its spelling mistakes are ones that we would want a Cree dictionary search system to completely ignore. We want our search system to be customizable to East Cree, and ideally to any other language.

Customization

In Section A, we described a variety of common spelling mistakes within the East Cree language. We used these to create a list of aspects of Cree spelling to include into our search system. We further refined it by distinguishing between spelling mistakes to completely ignore and spelling mistakes to partially ignore. In the first category (completely ignore), we included very common spelling mistakes and what can be considered alternate spellings. In the second category (partially ignore) we included what we considered to be greater violations of the orthography

For spelling differences that are completely ignored by the dictionary search system, we chose:

–voiced versus voiceless consonants (writing a *b* instead of a *p*, a *d* instead of a *t*, etc.)
–French versus English spelling differences (*j* instead of *sh*, *f* or *ph* instead of *v*)
–older spelling systems (*o* instead of *waau*, *ei* instead of *eyi*, *ai* instead of *ayi*)
–missing *h* in a word
–vowel length (*a* instead of *aa*, *i* instead of *ii*, etc.)

The first three of these spelling differences are uncontroversial; they simply indicate common alternate spellings, and do not lead to ambiguity. Indeed, they might be considered correct alternate orthographies, rather than actual spelling mistakes. The last two (missing *h* and vowel length) are clearly actual spelling errors. However, they occur commonly enough that we have chosen to treat the difference between *a* and *aa*, or having an *h* and not having an *h* as *no difference at all*. That is, when the user is searching for a word in the dictionary, and the word they have typed in does not exist, then the system should proceed by ignoring any differences involving vowel length, the presence of a small *h*, voiced versus voiceless consonants, and so on.

This decision will certainly lead to a slight over-generalization by the search system, due to the presence of certain minimal pairs within the Cree language. For example, a user searching for *siituu* but incorrectly spelling it *situ* would receive equally strong matches for *siituu* and *siihtuu*. However, this sort of

scenario at worst gives the user two options to choose from, and so we feel it is acceptable.

Certain other spelling errors, while still common, were deemed to not be completely ignorable. Changes in vowel quality or the loss of a *w* are common, but we should not go as far as to treat these differences as no difference at all (as we did with vowel length). However, these spelling mistakes are clearly not as large an error as changing a vowel to a consonant, or forgetting a whole letter. We thus developed a list of spelling mistakes that should be treated as *half* of an error. Inspired by the Soundex system, we also included relevant Cree phonemic-based errors and typos:

–vowel quality (*i* instead of *a*, *u* or *e*)
–missing *w*
–mistaking nasals (*m* instead of *n*)
–mistaking labials (*f* instead of *p*)
–typing *tch* for *ch* [ ]
–mistaking *s* for *sh* [ ]

We now have two lists of common spelling errors, one set we wish to treat as if they are not differences, and one set we wish to treat as if they are *half* a difference (in terms of the Levenshtein system). To implement this, we needed an algorithm that allows for both an efficient computation of the total difference, and allows us to easily adjust and modify these two lists of differences.

This last point is worth further highlighting. By allowing these two lists of common spelling errors to be easily modified, we are in fact making our search system customizable *to different languages*. The only aspect of our system which is specific to the East Cree language is our choice of errors. By adjusting these lists, we can make the system suitable for different languages. Furthermore, we can also make fine-tuning adjustments to these rules as more data are gathered on common East Cree search problems.

Algorithm

Given the extensive research that has gone into the Levenshtein Distance algorithm, making it fast and efficient, we decided to base our system around it. However, Levenshtein has no method for indicating *half* differences. This led us to develop the following algorithm.

First, we defined two functions which modify a given word based on the two lists of spelling variations identified in the previous section. These are called *relax1* and *relax2*. Each of these is meant to take a given word and produce an output which is guaranteed to be identical for any two words which only differ by errors indicated in the relevant list. For example, *relax1* will transform *naapeu*, *napeu*, *nabeu*, *naabeu*, *naahbeu*, and so on into *napeu*. As a further example, *relax2* would transform *napeu*, *mapiu*, *nepeu*, *nupiu*, *napheu*, *mipweu*, and so on into *nøpøø*. It should be noted that we are using *ø* to indicate a neutral vowel.

We now create two new lists of words, based on the words in the original dictionary. The first list (*words1*) consists of the result of running every word in the dictionary through *relax1* only. The second list (*words2*) is formed by running every word through both *relax1* and *relax2*. For example,

Original	***Words1***	***Words2***
naapeu	napeu	n*øpøø*
napet	napet	n*øpøt*
napeuu	napeu	n*øpøø*
iskweu	iskweu	*øskøø*

Now suppose a user enters a search word which is not in the main dictionary list. We first transform the word into *search1* (via *relax1*) and *search2* (via *relax1* and *relax2*). Next, we measure the Levenshtein distance between *search1* and all of the words in *words1*. This gives us a difference value that ignores any of the changes identified in the first list of spelling mistakes. We then measure the Levenshtein distance between *search2* and all of the

words in *words2*. This gives us a difference value that ignores any of the changes identified in both of the spelling mistake lists. To attain our final difference value, we simply take the average of these two differences. The result is a value which treats the differences in the second spelling mistake list as only half as important as the differences in the first list.

To optimize this process, we note that we do not need the values for every single item in the dictionary. Instead, there should be some threshold of error after which the word should be ignored. For example, when searching for close matches for *napeu*, the word *iskweu* would have much too large a difference to be considered. Most Levenshtein distance algorithms allow for the specification of a threshold of error above which the word should be ignored. This is usually specified relative to the length of the word (so as to allow for more errors in longer words). Thus, a threshold of 0.2 would indicate that at most one error (i.e. one insertion, deletion, or replacement) is allowed for every 5 letters in the original word.

If we specify this threshold, we can greatly speed up the search process. Suppose we only want to find words which are within a threshold 0.2 of the word the user has entered. To do this, we first do our initial measurement of the Levenshtein distance between *search1* and all the words in *words1* using a threshold of (0.2*2)=0.4. The reason we choose a threshold of 0.4 is that this value is going to be averaged with the result of the Levenshtein difference for the second list, and if the value for the first list is above 0.4, then the combined average cannot be below 0.2. This gives us a smaller set of words to work with for the second stage.

The following table shows an example of searching given the entry *napeu*:

Original	***Words1***	***Difference1 napeu***	***Words2***	***Difference2 nøpøø***	***Average Difference***
naapeu	napeu	0	n*øpøø*	0	0
napet	napet	0.2	n*øpøt*	0.2	0.2
naapeuu	napeu	0	n*øpøø*	0	0
iskweu	iskweu	0.8	*øskøø*	0.6	0.7

As we can see from this table, the dictionary words *naapeu* and *naapeuu* are determined to have a smaller average difference than *napet*. This is a desirable result since it is not reasonable to believe that a user means *napet* if they have entered *napeu*.

Given the simplicity of the above example, we can see that the first stage of the algorithm (Difference1) is able to determine that *naapeu* and *naapeuu* are closer matches than *napet*. In more complex situations, the second stage of the algorithm is needed. This can be seen in the following table, showing an example of searching given the entry *mwiikw*:

Original	***Words1***	***Difference1 mwikw***	***Words2***	***Difference2 møk***	***Average Difference***
mwaakw	mwakw	0.2	m*øk*	0	0.1
wiikw	wikw	0.2	*øk*	0.33	0.266
amihkw	amikw	0.4	*ømøk*	0.33	0.366
maak	mak	0.6	*møk*	0	0.3

In this case, the first stage (Difference1) indicates that both *mwaakw* and *wiikw* are identically close to *mwiikw* (a difference of 0.2). The second stage (Difference2) finds that *mwaakw* is much closer than *wiikw* (a difference of 0 versus 0.33). This is because the second stage treats a consonant drop as a worse violation than a vowel quality change. However, the second stage is also somewhat over-zealous, as it indicates that *maak* is just as close to *mwiikw* as *mwaakw* is. For this reason, we average the results of

Difference1 and Difference2, resulting in *mwaakw* being correctly chosen as the most likely word.

Implementation

Since our system is meant to be easily customizable (both for adjusting the results to take into account different sorts of spelling errors and for switching to completely different languages), it is worthwhile to describe the actual software implementation of the two functions relax1 and relax2 which are at the heart of the algorithm.

The first function, *relax1*, is meant to take a given word and modify it so that all of the spelling variations described in our first list (errors which should be considered to have a difference of 0) disappear. This function, implemented in the Python programming language, looks like this:

```
def relax1(word):
  word=word.replace('aa','a')          # vowel length
  word=word.replace('ii','i')          # (or top dot on syllabics)
  word=word.replace('uu','u')

  word=word.replace('b','p')           # voice/voiceless consonants
  word=word.replace('d','t')
  word=word.replace('g','k')
  word=word.replace('z','s')
  word=word.replace('v','f')

  word=word.replace('j','sh')          # french/english spelling
  word=word.replace('ph','f')
  word=word.replace('tch','ch')
  word=word.replace('dj','ch')

  word=re.sub('(?<!c|s|t)h','',word)   # remove 'h' (except in ch,
                                       #  sh, and th)
  word=word.replace('o','wau')         # older spellings
  word=word.replace('ei','eyi')
  word=word.replace('ai','ayi')

  return word
```

These are all commands which replace all instances of a particular pattern with another pattern. With one exception, these are simple patterns, such as replacing *aa* with *a*. A more complex pattern is used in the one situation where we want to remove all *h*'s except for the ones after a *c*, *s*, or *t*. This is accomplished using a *regular expression* (a powerful matching system built in to most programming languages).

The second function, *relax2*, carries out the elimination of the remainder of the spelling changes.

```
def relax2(word):
  word=word.replace('i','ø')          # merge all vowels into ø
  word=word.replace('u','ø')
  word=word.replace('e','ø')
  word=word.replace('a','ø')

  word=word.replace('m','n')          # merge nasals (Soundex inspired)
  word=word.replace('f','p')          # merge labials (Soundex inspired)

  word=word.replace('ch','sh')        # french/english spelling

  word=word.replace('sh','s')         # East Cree dialectal difference


  word=word.replace('w','')           # remove w's (or left dot on syllabics)

  return word
```

By merely adjusting these two functions, the entire search system can be adjusted for different conditions.

VERB INFLECTION SEARCH

While the relaxed search system described in the previous section makes searching for words in the dictionary a much more forgiving process, this requires that the correctly spelled word be in the dictionary. However, for Cree verbs, this is an impossibility due to the large number of inflected forms for each verb. Instead, only the basic stem form of the verb is normally given. This

means that if a user searches for “nimishikaan” (I arrive by canoe), this will not be found, since the dictionary only contains the form “mishikaau”.

To address this difficulty, we have developed a system that can analyze the entered verb, identify the stem, and identify the verb form. This system makes use of the relaxed search, so it is able to identify even misspelled verbs. It provides a list of possible stems and verb forms, sorted so that the most likely candidates (taking into account Cree spelling patterns) are given first.

Verb paradigms database

Since this system must identify verb forms, it must have an example of each possible Cree verb form. This is provided by the Verb Paradigms Database[5]. This is a standard Toolbox database that provides an attested example of each inflection for each paradigm and stem type. For example, the entry for *nimishikaan* contains the following information:

Form	nimishikaan
Part of Speech	VAI
Paradigm	01
Person	1
Stem Type	long aa
Canonical Form	mishikaau
Gloss	I arrive by canoe

To facilitate the search system’s algorithm, one further field was added to the database. This was meant to mark the prefix and suffix information that identifies the inflected verb form. For *nimishikaan*, this would be marked as

ni	mishik	aan

Note that from a morphological perspective, the stem is *mishikaa-* and the suffix is *-n*. Our decomposition does not always follow Algonquian word formation rules but rather often includes in what we call the 'suffix' the final vowel or consonant of the stem. The

idea here is that the prefix *ni* and the 'suffix' *aan* are what the search system should look for when determining whether a verb is in this paradigm. These should be as long as possible while still being common to all verbs following this example. In this case, this means that all VAI verbs that are of the long aa stem type will start with “ni” and end with “aan” when inflected as paradigm 01 in person 1[6]. This will be used by the search algorithm to identify other verbs in this form. It should be noted that since the algorithm uses the relaxed search system, it will still identify verbs correctly even if the prefix and/or suffix are slightly misspelled.

Determining the paradigm

Given the database of verb paradigms described in the above section, the search system must first identify which paradigms could have been intended by the user. This is done by examining the beginning and ending of the entered verb and matching them to all of the suffixes and prefixes in the paradigm database. This is done by going through the complete list of paradigms and using the relaxed matching system (as described previously).

For example, if the misspelled word *nipimipahtaanawaa* is entered, then the system would find a very close match for the paradigm that starts with *ni* and ends with *aanaawaa* (VAI, paradigm 02, person 1, long *aa* stem). However, it will also find a close match for other paradigms, some of which are shown in the following table.

Part of Speech	Paradigm	Person	Stem Type	Prefix	Suffix
VAI	02	1	long aa	ni	aanaawaa
VTI	02	1	im	ni	aanaawaa
VTA	02	1-3	consonant	ni	aawaa
VAI	03a	3p	long aa		aahtaawaau

Given this approach, we can find a small set of potentially matching paradigms. This list can be made longer or shorter depending on how close the matches are required to be. For

example, the last paradigm shown in the above list matches the expected suffix *aahtaawaau*, even though the original word entered was *nipimipahtaanawaa*.

Determining the verb

Given this list of potential matching paradigms, the algorithm must next attempt to determine the actual verb stem intended. To do this, we make use of the same Cree dictionary as in the normal relaxed search dictionary system. Recall that this dictionary only contains verbs in their standard canonical form.

To determine the intended verb, the algorithm starts with the portion of the entered word that is left over once the suffix and prefix are removed. It then attempts to *reconstruct* the correct form. This is done by examining the original paradigm example. For example, the paradigm example for *ishki nimishikaanaawaa* contains the following information

Form	ishki nimishikaanaawaa
Part of Speech	VAI
Paradigm	02
Person	1
Stem Type	long aa
Canonical Form	mishikaau
Gloss	It seems that I am arriving by canoe
Prefix	ni
Suffix	aanaawaa

The search algorithm determines that in order to convert *nimishikaanaawaa* (the inflected form) to *mishikaau*, we must remove the prefix and suffix, and then add *aau*. This same rule can then be applied to the text that the user has entered. For the misspelled word *nipimipahtaanawaa* (given in the previous section), once the above paradigm example is identified as a potential match, the search system attempts to identify the verb by following the same rule: the prefix and suffix are removed, and -*aau* is added. This results in *pimipahtaau*. This text can then be

used to search the standard dictionary, and the closest match is *pimipihtwaau* 'to run carrying something'.

This process must be performed for each of the paradigm examples identified as potential matches. Each one may provide a different possible verb. For example, as shown in the previous section, another possible paradigm match for *nipimipahtaanawaa* is as follows:

Form	ishki nitihtutaanaawaa
Part of Speech	VTI
Paradigm	02
Person	1
Stem Type	im
Canonical Form	ihtutim
Gloss	It seems that I am doing it
Prefix	ni
Suffix	aanaawaa

For this case, the inflected form is converted to the canonical form by removing the prefix and suffix and then adding *im*. Thus, if this algorithm is applied to *nipimipahtaanawaa*, we create the potential stem *pimipahtim*. Searching through the dictionary for a matching verb results in *piimipitim* 'to turn something'.

Following this process for each of the potential paradigm matches gives a long list of potential verbs and verb forms. This list can be sorted based on the closeness of the match (the number of "errors", as determined using the relaxed matching algorithm discussed above). This is what our verb search system does, and then presents the user with the resulting sorted list.

Ongoing work

Our verb inflection search system is still in its initial stages. We are currently experimenting with different ways of presenting the resulting information and incorporating the verb dictionary into the on-line dictionary (http://dict.eastcree.org)

We have also been working on a special addition to the algorithm to deal with initial change. In some paradigms, the initial

syllable of the verb stem is always changed from its base form. Some paradigms, like the Conjunct Indicative (#11) uses initial change for questions. Which other paradigms allow for initial changes still needs to be fully documented. Forms with initial change do not depend on stem information, found in the Verb Paradigms database, but rather on phonological rules affecting the intial of any verb found in the (word) Dictionary. To allow matching of a changed verb with the dictionary citation form, changed forms must be either entered manually into the (word) Dictionary, or generated algorithmically. We are currently exploring the latter solution, by generating changed forms for the 15 171 verbs contained in the East Cree dictionary (Northern dialect), and checking with speakers whether the changed forms so generated matches their intuition.

Finally, although our system is capable of handling the most common preverbs like past *chiih*, volitional *wiih*, or conjunct preverbs like *aah* or *kaa*, we wish to expand this capability to handle the full set of Cree preverbs. But this requires additional documentation work on possible co-ocurrences between preverbs and paradigms.

ADAPTING THESE TOOLS TO THE INNU LANGUAGE

Our search tool can be adapted to other Algonquian languages, if common spelling errors or alternative spelling usages are well identified. We adapted the relaxed search engine to the Innu language recently[7]. For example, Innu standard orthography does not indicate vowel length, but has similar voice/voiceless confusion problems. Without going into as many details, we give below the two functions *relax1* and *relax 2*, implemented in the Python programming language, as we defined them for the Innu on-line dictionary (www.innu-aimun.ca/lex), to illustrate how such an adaptation is possible.

```
def relax1(word):

 word=word.replace('b','p')                    # voiced/voiceless consonants
  word=word.replace('d','t')
  word=word.replace('g','k')
  word=word.replace('z','s')
  word=word.replace('j','sh')
  word=word.replace('v','f')

  if word.endswith('w'): word=word[:-1]        # forget final raised u

  word=re.sub('(?<!c|s)h','',word)             # remove 'h' (except in ch, sh)

  return word
```

```
def relax2(word):

  word=word.replace('i','a')            # mixing up short vowels
  word=word.replace('u','a')            (includes ai->ei, ei->ai, etc)
  word=word.replace('e','a')

  word=word.replace('nat','nt')         # Forget vowels between n and t
  word=word.replace('tan','tn')

  if word.endswith('uan'):              # write -un for -uan at end of words
    word=word[:-3]+'un'

  # forget h before a consonant
  #  Note that this also handles st->sht, sp->shp, sk->shk
  #  Also note that relax1 currently gets rid of any 'h's that aren't
  #  in ch or sh
  word=re.sub('h(?![rtpshklcnm])','',word)

  word=word.replace('ss','s')  # handles writing 's' instead of 'ss'

  word=word.replace('ch','sh')  # also handles tsh->tch
  word=word.replace('tsh','ts')

  return word
```

Similarly to what we did for the Cree verb dictionary, the Innu verb dictionary (adapted from Baraby, 2004) will include the Innu search engine above.

CONCLUSION

We have shown here how we approached the development of a search engine for the Eastern James Bay Cree dictionary on the web (dict.eastcree.org) in order to allow spelling mistakes or creative spelling. We demonstrated that this tool was in turn easily adaptable for a neighbouring Algonquian language, namely Innu. We also showed how it was used to help relate inflected forms to citation forms in a dictionary.

To build a successful tool, we adopted an interdisciplinary approach, combining descriptive linguistics, usage observation and computational creativity. Rather than just copy Soundex or other English-biased approaches to spell-checking, we had to first look at the rules of the Cree language. The fact that it uses a syllabic writing system offered an additional challenge, solved by implementing systematic conversions from roman to syllabic orthographies. In order to customize our search engine for Cree, a good linguistic description of the language was necessary, as well as observations of spelling patterns of Cree users. On the computing side, we combined two approaches to spell-checking: measuring the difference between the word typed and the words in the dictionary, and phonetic matching. The originality of our computational system is in allowing half-errors in Levenshtein differences, allowing for more accurate results. Our algorithm is fast and can be easily added to other existing systems, because its basic components exist in most modern programming languages.

Using our results, other kinds of research tools could be created. For example, a system that, given a text of unknown provenance, could estimate the probability that it is a sample of East Cree that could be made available on the eastcree.org website, so that archivists, historians, genealogists, lawyers, and other researchers could copy and paste unidentified text into a box a

click a 'test' button. The same system could be used as the basis for a more general Algonquian language identifier.[8]

The question remains, however, as to whether our tools will be used and useful in accomplishing our goal in helping preserve a minority language. For that we can study the patterns of usage of the dictionary on the web, recording the language of queries. So far, data on on-line usage indicate that the Dictionary Database is queried around 36% of the time in Cree, with an average of over 2,000 queries per month. The charts in Appendix B give the usage data for the dictionary for the year 2007 and for an earlier period (2005, 2006). Usage of the on-line Cree Dictionary has gone up in absolute terms from slightly over 11,000 queries in 2006 to nearly 25,000 queries in 2007. Queried languages are stable, with the addition of French since the publication of the French version in April 2007. An almost equal number of queries are using the Cree syllabics and the Cree roman orthography. Our next step will be to observe how and when the relaxed search is being used. For that we are starting to record data on the relaxed search, in addition to data on the regular queries per language.

We can expect that the more the language is endangered, the more the searches will be performed in the colonial languages (English or French). However, when a word is searched for in the native language, the discouragement that users experience in not finding the word they are looking for could even accelerate the dominance of a colonial language. Thus, a tool that allows alternative spelling, like the one described here, should help balance this tendency for dominance.

As we do not wish our electronic dictionaries to end up having the reverse effect of the one anticipated (i.e. accelerating language loss instead of promoting retention) we hope that the tool described here will have a positive impact for language preservation. We also hope that this first-of-its-kind search engine will serve as a model for dictionaries of other Algonquian, endangered or minority languages.

APPENDIX A: EASTERN JAMES BAY CREE SYLLABIC CHART

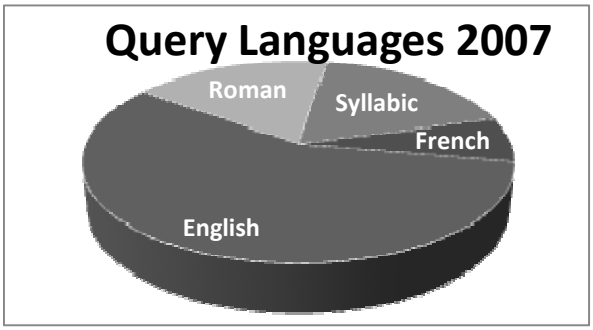
Finals

	∇ E	△ i	△̇ ii	▷ u	▷̇ uu	◁ a	◁̇ aa			° u	ˮ h
	·∇ We	·△ wi	·△̇ wii	·▷ wu	·▷̇ wuu	·◁ wa	·◁̇ waa				
∨ pe	·∨ pwe	∧ pi	∧̇ pii	> pu	>̇ puu	< pa	<̇ paa	·<̇ pwaa		ᐸ p	
∪ te	·∪ twe	∩ ti	∩̇ tii	ↄ tu	ↄ̇ tuu	ᑕ ta	ᑕ̇ taa	·ᑕ̇ twaa		ᑦ t	
ϥ ke	·ϥ kwe	ᑭ ki	ᑭ̇ kii	ᑯ ku	ᑯ̇ kuu	ᑲ ka	ᑲ̇ kaa	·ᑲ̇ kwaa		ᑳ k	ᑴ kw
ᒉ che	·ᒉ chwe	ᒋ chi	ᒋ̇ chii	ᒍ chu	ᒍ̇ chuu	ᒐ cha	ᒐ̇ chaa	·ᒐ̇ chwaa		ᒡ ch	
ᒣ me	·ᒣ mwe	ᒥ mi	ᒥ̇ mii	ᒧ mu	ᒧ̇ muu	ᒪ ma	maa	· mwaa		ᒫ m	
ᓂ ne	·ᓂ nwe	ᓇ ni	ᓇ̇ nii	ᓄ nu	ᓄ̇ nuu	ᓈ na	ᓈ̇ naa	·ᓈ̇ nwaa		ᓐ n	
ᓗ le	·ᓗ lwe	ᓕ li	ᓕ̇ lii	ᓘ lu	ᓘ̇ luu	ᓚ la	ᓚ̇ laa	·ᓚ̇ lwaa		ᓪ l	
ᔦ se	·ᔦ swe	ᔨ si	ᔨ̇ sii	ᔩ su	ᔩ̇ suu	ᔦ sa	ᔦ̇ saa	·ᔦ̇ swaa		ᔭ s	
ᔿ she	·ᔿ shwe	ᔾ shi	ᔾ̇ shii	ᕁ shu	ᕁ̇ shuu	ᔕ sha	ᔕ̇ shaa	·ᔕ̇ shwaa		ᔖ sh	
ᕀ ye	·ᕀ ywe	ᕂ yi	ᕂ̇ yii	ᕃ yu	ᕃ̇ yuu	ᕆ ya	ᕆ̇ yaa	·ᕆ̇ ywaa		ᕇ y	
ᖇ re	·ᖇ rwe	ᖏ ri	ᖏ̇ rii	ᖅ ru	ᖅ̇ ruu	ᖃ ra	ᖃ̇ raa	·ᖃ̇ rwaa		ᖄ r	
ᐯ ve	·ᐯ vwe	ᐁ vi	ᐁ̇ vii	ᐅ vu	ᐅ̇ vuu	ᐊ va	ᐊ̇ vaa	·ᐊ̇ vwaa		ᐋ v, f,	
ᐄ the	·ᐄ thwe	ᐃ thi	ᐃ̇ thii	ᐁ thu	ᐁ̇ thuu	ᐧ tha	ᐧ̇ thaa	·ᐧ̇ thwaa		ᐦ th	

APPENDIX B: QUERY DATA

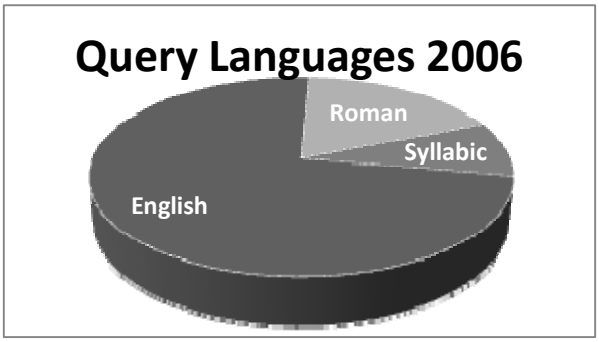
Query Languages 2007 (Jan-Dec)

Total	**24757**	**100%**
English	14159	**57%**
Roman	4344	**18%**
Syllabic	4496	**18%**
French	1758	**7%**



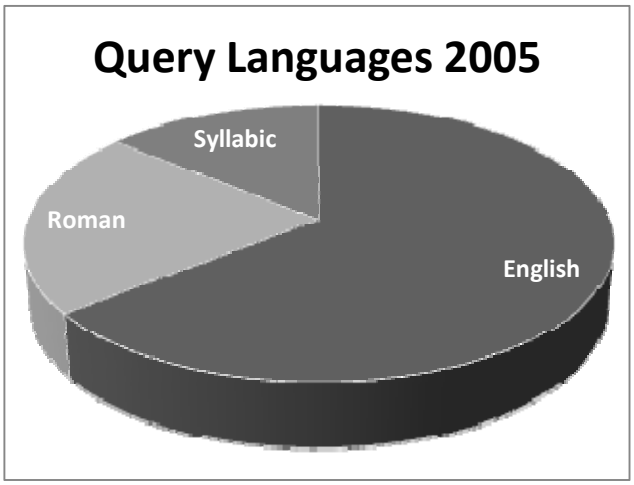
Query Languages 2006 (Jan–Dec)

Total	**11338**	**100%**
English	8270	**73%**
Roman	2055	**18%**
Syllabic	1013	**9%**



Query Languages 2005 (May-Dec)

Total	**7836**	**100%**
English	5038	**64%**
Roman	1699	**22%**
Syllabic	1099	**14%**



ENDNOTES

[1] This work was made possible through a SSHRC grant (# 856-2004-1028). Adaptation to Innu was partially subsidized by SSHRC CURA grant (# 833-2004-1033). We wish to thank the audiences at the 37th and 39th Algonquian Conferences, as well as Bill Jancewicz, Marguerite MacKenzie, Rand Valentine and our anonymous reviewers for comments and suggestions. Thanks to Fred Mailhot and Delasie Torkornoo for help in retrieving the Cree Dictionary usage statistics. Chinaskumitinwaau to the many Cree speakers, to Cree program staff members and to our students for testing the relaxed search. The adaptation to Innu would not have been possible without José Mailhot's expertise with Innu language and orthography. All remaining errors are ours.

[2] The East Cree syllabic fonts are available at resources.eastcree.org. A chart explaining the roman-syllabic correspondance is given in Appendix A.

[3] We have so far focussed on inflected verbs rather than inflected nouns, for three reasons: First, verbs are by far the largest part of the dictionary (and also the most frequent in language use). Second, noun inflection usually consists of

fewer syllables than verb inflection and is quite well handled already by the search engine based on spelling rules. Third, dependent nouns and inflected pronouns have been given entries in the Cree dictionary. In the future, we plan to include the rules for possessed nouns into our search.

[4] We are using the numbering system for paradigms developed by MacKenzie (1992) that allows comparison of paradigms between the Cree-Innu (Montagnais-)Naskapi dialect continuum.

[5] The verb paradigm documentation for East Cree was started by Marguerite MacKenzie (1980), who then worked for many years with the Cree School Board to document the verb forms, leading to the production of several unpublished work copies over the years. The Database structure in Toolbox was modelled after the one Bill Jancewicz developed for the neighboring language Naskapi. The Verb paradigm database (Northern) used here is the latest version (2008) of the one by Junker, MacKenzie and Salt (2002).

[6] Combination with preverbs to which the personal prefix attaches are dealt with as well.

[7] José Mailhot, who has been involved with the entire standardization process of the Innu orthography, provided us with a list of common difficulties encountered by Innu writers.

[8] We are grateful to an anonymous reviewer for this suggestion.

REFERENCES

Baraby, Anne-Marie. 2004. *Guide de Conjugaisons en langue innue*. 2. Sept-Iles: Institut culturel et educatif montagnais.

Hamming, Richard W. 1950. Error-detecting and error-correcting codes. *Bell System Technical Journal* 29.2:147-160.

Junker, Marie-Odile (ed.) 2000-2008. *The East Cree language web.* www.eastcree.org.

Junker, M.-O., M. MacKenzie, L. Salt, A. Duff, D. Moar & R. Salt (eds). 2007-2008. *Dictionnaire du cri de l'Est de la Baie James sur la toile: français-cri et cri-français (dialectes du Sud et du Nord).* http://dictf.eastcree.org/

Junker, M.-O., M. MacKenzie & L. Salt. 2002. *East Cree verb paradigms (Northern dialect):* A database in Toolbox, complete with sounds. Last revised in January 2008. Unpublished.

Levenshtein, Valdimir I. 1965. Binary codes capable of correcting deletions, insertions and reversals. *Doklady Akademii Nauk SSSR* 163.4:845-848.

MacKenzie, M., Duff, E., B. Jancewicz, M.-O. Junker, D. Moar, E. Neeposh, L. Bobbish-Salt, & R. Salt. 2004-2008. *The Eastern James Bay Cree Dictionary on the Web : English-Cree and Cree-English (Northern and Southern dialects).* http://dict.eastcree.org/

MacKenzie, Marguerite, Annie Whiskeychan, Daisy Moar, Ruth Salt & Ella Neeposh. 2004. *The East Cree Spelling Manual (Southern dialect)*, Cree School Board:available on-line at: www.resources.eastcree.org

MacKenzie, Marguerite. 1992. *Verb Paradigms in East Cree and Naskapi*. 24th Algonquian Conference Tertiary Verb Paradigms in East Cree and Naskapi. Carleton University

Russell, Robert. *Soundex.* US Patent 1261167, 1918.

Salt, Luci Bobbish and Marguerite MacKenzie. 2006 *The East Cree Spelling Manual (Northern dialect).* Cree School Board: www.resources.eastcree.org