Language technology for endangered languages: Sámi as a case study

Trond Trosterud
Faculty of the humanities
University of Tromsø

October 16, 2006

Contents

1 Introduction

Seen from a purely computational point of view, all languages pose the same challenges for computational linguists, and there is no reason to treat endangered and non-endangered languages differently. When it comes to doing language technology in practice, the situation is different. There is no economical demand to make language technology solutions for more than a handful of languages. For most languages the basic tools for making language technology applications are not readily available: there are not large amounts of texts available in electronic format, also reference grammars may be incomplete. On the positive side, we need not repeat costly mistakes made by the lg tech pioneers. Projects starting today may build clean, modern systems.

Without well-developed language technology resources, no language will in the future be able to:

function as an administrative language. (Text must be proofread, different types of schemes and fill-in forms must be generated, people will need summaries and abstracts, all this will be done automatically, with the help of language technology tools.)

function in a bilingual administration (multilingual versions of the same text will be generated by means of machine translation, consistent use of multilingual terminology will be checked automatically)

be stored in digital archives (Source engines will rely upon language technology in order to classify documents and information, and in order to find stored text. We would also like to search for content, independent of what language the document is written in)

When witnessing the benefits human-machine interaction may give, people will find very strong reasons for preferring English and other majority languages to languages where no language technology is available. After all, why store a document written in a language which makes it impossible to retrieve?

In short, if we do not capitalise upon language technology for minority languages, their users will soon feel themselves in the same situation as the left-handed witnessing a right-handed person using a pair of scissors. During the last decade, great progress has been made within the area of semantic nets and word-sense disambiguation. This work still hasn't emerged on the average user's desktop, but it will, and unless we don't act, it will emerge for the large languages only. As commercial language technology products become better, and as public administration relies more heavily upon language technology, the difference between languages with and languages without these resources will become strikingly clear.

In addition to the practical concerns involved, we linguists have our own axe to grind here: A well-developed language technology will make it possible to make the language in question an object of study in another and more efficient way than without it.

This is actually a very important point: For most of the languages of the world, there is no commercial incitement to do the ground work needed for language technology applications. What is left is then academic research institutions. The professional linguist sees the writing of comprehensive reference grammars for as many languages as possible, ultimately for every language of the world, as a goal in itself.

Computers manipulate bits and bytes. Whenever the byte manipulation takes into account what language it manipulates, we deal with language technology. Here, I will concentrate upon text-based language technology. I will have a look at the localisation, the basic infrastructure for writing a language on computers, and I will look at basic grammatical analysis tools, such as morphological parsers and disambiguators. I will also briefly look into semantical and multilingual language tools, as well as certain subcomponents of text-to-speech programs.

In this article, I will argue that the availability of language technology tools will be of utmost importance in order to be able to use a language as administrative language. I will also present some prerequisites for language technology work, and some paths for the work ahead.

The article is structured as follows: First comes a discussion of different types of text-based language technology. Thereafter we look at some case studies, above all Northern Sámi, but also other languages. The focus will be upon localisation work, and upon how grammar-based parser construction can be used as a basis for reference work. Finally comes a conclusion.

2 Different types of text-based language technology

Language technology development is paid for partly by the industry, and partly in publicly funded research institutions. As a matter of fact, many language technology products for the commercially most interesting languages are developed by public funding, in academic research institutions. Much of the work done for English is even paid for by governments in non-anglophone countries.

2.1 Localisation

By localisation, we mean everything that makes the computer aware of what language the user is using, and what country he or she lives in: character sets, keyboard layout, sorting order, day-and-time format, currency symbol, and many other things. The language-independent infrastructure is provided by international standard organisations, and tools for making language-specific versions are in most cases available.

The 1990ies witnessed the largest revolution in language typesetting since Gutenberg: All the letters of all the languages of the world, living and dead, were made unambiguously available on every computer, by being included in the character set standard ISO/IEC 10648, or Unicode. If any letter is missing, it will be included in Unicode.

At present, there are three major operative systems competing for the PC market: Windows, Macintosh and Linux. Note that, from the point of view of minority languages, Linux occupies a special position, as it is made on a volunteer basis. If someone needs a keyboard layout for a hitherto unsupported language, they can just go ahead and make the layout, and then include it in the Linux desktop distribution, so that it becomes available to everyone. For Windows and Macintosh, the situation is different: These are closed operating systems, and the adding of language resources must be approved by Microsoft and Apple, respectively.

2.2 Morphological parser and disambiguator

Unfortunately, so far, most language technology applications have been made for commercially interesting languages. This means that, although Unicode is the standard on web browsers, language technology source code is, as a rule, written in some 8-bit codepage, or even in 7-bit ASCII.

Many language technology applications for English have been done with a minimum amount of grammatical analysis. Spell checkers have been made base upon a compressed list of all wordforms found in a large corpus of proofread running text.

Two factors have contributed to the relative success of this approach: First, there are very many texts electronically available for English. Second, English has a very poor morphological structure, where each lexeme is represented by at most a handful of wordforms, with no forms significantly less frequent than the others. Most of the languages of the world have a more elaborate word structure than English, with large paradigms for each lexeme, containing tens or hundreds of wordforms. In these languages, some of the paradigm members have a very low text frequency. Statistical approaches work best for languages with huge amounts of text electronically available, and a minimal amount of morphology. Grammatical analysers, will analyse and generate every theoretically possible inflection form. If they are not properly restricted, they will, however, run the risk of generating nonexistent forms, this risk is eliminated in the list-based approach.

In order to illuminate thedistinction between these two approaches, I conducted an investigation on verbs in English, Danish, Sámi and Finnish. The starting point was the verbs in figure 1, a high- and a mid-frequency verb.

	Corpus type	Words	High-frequency verb	Mid-frequency verb
Sámi	New Testament	145282	*dadjat*	*bálvalit*
Danish	New Testament	177051	*sige*	*tjene*
English	New Testament	188616	*say*	*serve*

Figure 1: Check of coverage for high- and mid-frequency verb

Focusing first on English and Danish, we get the following pattern. Figure 2 shows how many times each word form of the four different verbs are found in the New Testament. Word forms not found are marked with a grey shade.

	Danish					English				
Inf	**sige**	272	**tjene**	37		Inf	**say**	422	**serve**	33
Prs	**siger**	627	**tjener**	16		Pst	**said**	1058	**served**	5
Pst	**sagde**	1289	**tjente**	9		Ger	**saying**	408	**serving**	6
PftPtc	**sagt**	125	tjent	0		2Sg			**serveth**	5
Imp	**sig**	860	tjen	0						
PrsPrt	**sigende**	4	**tjenende**	2						

Figure 2: Attested verbforms in the Danish and English New Testament

The English -s forms serves and says are missing because this version of the New Testament (King James' Bible) is from 1611, and it contains no -s forms. Looking for says and serves in an arbitrary corpus of 85000 words , gives 4 says and 2 serves. Except for the missing imperative and perfect participle of the Danish mid-frequency verb, all inflected forms are thus covered.

Let us then have a look at Sámi. A North Sámi verb has 45 core finite forms (+ some marginal ones), and a dozen or so infinite forms. In addition come derivational forms, passive, causative, inchoative and others, each derivational process will give rise to a new set of inflected forms. Here I will concentrate only upon the core forms. The wordforms of *dadjat* 'to say' found in the NT, are shown in figure 3

As can be seen from Table 6, The New Testament contains only 19 distinct forms of *dadjat*, or 39 % of the core forms. The coverage is best for Indicative Present, but also this subparadigm is incomplete.

Northern Sámi *bálvalit* 'to serve', in the NT, is found in figure 4. For *bálvalit*, only 12 distinct forms are found in The New Testament, or 25 % of the actual word forms.

	Ind.Present		Ind.Past		Cond.Pres		Pot.Pres		Imperative	
Sg1	**dajan**	4	dadjen		**dajašin**	1	dajažan		dadjon	
Sg2	**dajat**	6	**dadjet**	48	dajašit		dajažat		**daja**	=
Sg3	**dadjá**	42	**dajai**	207	**dajašii**	4	**dajaš(a)**	1	dadjos	
Du1	**dadje**	=	dajaime		dajašeimme		dajažetne		daddju	
Du2	**dadjabeahtti**	1	dajaide		dajašeidde		dajažeahppi		daddji	
Du3	dadjaba		**dajaiga**	7	dajašeigga		dajažeaba		dadjoska	
Pl1	**dadjat**	=	**dajaimet**	1	**dajašeimmet**		dajažit		dadjot	
Pl2	**dadjabehtet**	27	dajaidet		**dajašeiddet**	2	dajažehpet		daddjet	
Pl3	**dadjet**	=	**dadje**	183	dajaše(dje)		dajažit		dadjoset	
Neg	**daja**	4			**dajaše**	1	dajaš		**daja**	=
	Inf		PrsPrc		PrfPrc		VerbAbessive		Gerund	
	dadjat	47	**daddji**	4	**dadjan**	15	dajakeahttá		dajadettiin	

Figure 3: Northern Sámi *dadjat* 'to say', in the NT

	Ind.Present		Ind.Past		Cond.Pres		Pot.Pres		Imperative	
Sg1	**bálvalan**	=	Bálvalin		bálvaleačča		bálvalivčče		bálvalehkon	
Sg2	bálvalat		**bálvalit**	=	bálvaleačča		bálvalivčče		**bálval**	=
Sg3	**bálvala**	10	**bálvalii**	3	bálvaleažžá,-aš,-š		bálvalivčči		**bálvalehkos**	2
Du1	bálvaletne		**bálvaleimme**	1	bálvaležže		bálvalivččiime		bálvaleadnu,hkku	
Du2	bálvaleahppi		Bálvaleidde		bálvaleažžabeahtti		bálvalivččiide		bálvaleahkki	
Du3	bálvaleaba		Bálvaleigga		bálvaleažžaba		bálvalivččiiga		bálvalehkoska	
Pl1	**bálvalit**	=	bálvaleimmet		bálvaleažžat		bálvalivččiimet		bálvaleadnot	
Pl2	**bálvalehpet**	4	**bálvaleiddet**	1	bálvaleažžabehtet		bálvalivččiidet		**bálvaleahkket/ot**	2
Pl3	**bálvalit**	=	**bálvaledje**	2	bálvaležžet		bálvalivčče		bálvalehkoset	
Neg	**bálval**	5			bálvale(a)š, -čča,		bálvalivčče		**bálval**	=
	Inf		PrsPrc		PrfPrc		VerbAbessive		Gerund	
	bálvalit	39	**bálvaleaddji**	72	**bálvalan**	13	bálvalkeahttá		bálvalettiin	

Figure 4: Northern Sámi *bálvalit* 'to serve', in the NT

149000 words is a small corpus. In order to enlarge it, we must switch language. Finnish has a verbal morphology quote similar to the Sámi,, and it is represented by approcimately 3.5 billion words on the Internet (october 2004). The pendant to *bálvalit* is *palvella*, a verb among the 1000 most common Finnish words. Looking at the representation of its finite forms on the Internet, we find the pattern in figure 5. It is well represented on the net, as all its finite forms are attested (although to a varying degree).

	Ind.Present		Ind.Past		Cond.Pres		Pot.Pres		Imperative	
Sg1	**palvelen**	931	**palvelin**	56300	**palvelisin**	69	**palvellen**	2060		
Sg2	**palvelet**	423	**palvelit**	63	**palvelisit**	22	**palvellet**	11	**palvele**	8370
Sg3	**palvelee**	94300	**palveli**	8650	**palvelisi**	6090	**palvellee**	44	**palvelkoon**	199
Pl1	**palvelemme**	28100	**palvelimme**	114	**palvelisimme**	160	**palvellemme**	3	**palvelkaamme**	85
Pl2	**palvelettr**	413	**palvelittr**	88	**palvelisittr**	37	**palvellettr**	2	**palvelkaa**	960
Pl3	**palvelevat**	38700	**palvelivat**	2340	**palvelisivat**	1500	**palvellevat**	9	**palvelkoot**	94

Figure 5: Finnish *palvella* 'to serve' on the Internet

palvella is still a common verb, and on the net it is typically found in contexts like "our shop is open (and serves its customers) between 9 and 5). If we instead take a rare verb, we get a more meagre picture. Let us consider *vapisuttaa* 'make shake, shiver' a verb among the 12000 most common Finnish words, cf. figure 6.

11536	Ind.Present		Ind.past		Cond.Pres		Pot.Pres		Imperative	
Sg1	**vapisutan**	2	vapisutin	0	vapisuttaisin	0	vapisuttanen	0		
Sg2	vapisutat	0	vapisutit	0	vapisuttaisit	0	vapisuttanet	0	**vapisuta**	7
Sg3	**vapisuttaa**	122	**vapisutti**	83	**vapisuttaisi**	1	vapisuttanee	0	vapisuttakoon	0
Pl1	vapisutamme	0	vapisutimme	0	vapisuttaisimme	0	vapisuttaneme	0	vapisuttakaamme	0
Pl2	vapisutatte	0	vapisutitte	0	vapisuttaisitte	0	vapisuttanette	0	vapisuttakaa	0
Pl3	**vapisuttavat**	6	**vapisuttivat**	13	vapisuttaisivat	0	**vapisuttanevat**	1	vapisuttakoot	0

Figure 6: Finnish *vapisuttaa* 'make shake, shiver', on the Internet

As we can see from figure 6, even in a 3.5 billion word corpus, we are not able to fill more than 8 of 29 forms, or 27 %. And this is the core finite paradigm. If we were to take the full Finnish verb paradigm into account (which is 852 finite forms and infinitives, and 11000 participles), the percentage obviously would drop further.

A more reliable way of covering all the inflected form is a morphological transducer. A simple transducer is shown in figure 7. Combining a dictionary with declension class info plus a morphological transducer will give a grammatical analyser, capable of analysing running text. Such transducers can be written within a year or so, dependent upon the complexity of the language in question. One might also use machine learning to construct transducers, as recently investigated by [?], but that is not the topic of this article.

A possible counter argument to the use of transducers may be that the paradigms of verbs like *vapisuttaa* (and indeed of all other verbs) may be generated, either by a transducer, or by other morphological means, and then imported into some list-based language technology software. This is indeed possible, and there is nothing wrong with that, except for the fact that this workaround presupposes the transducer that it set out to avoid. As soon as we have a transducer for the language in question, we are able to choose, though. Either we can generate a fullform list as input to list-based applications, or we can use the transducer as such, and analyse or generate word-forms on the fly.

In order to make a morphological transducer, the following resources are needed:

1. A dictionary in electronic format, preferably with information on stem- and declension classes.

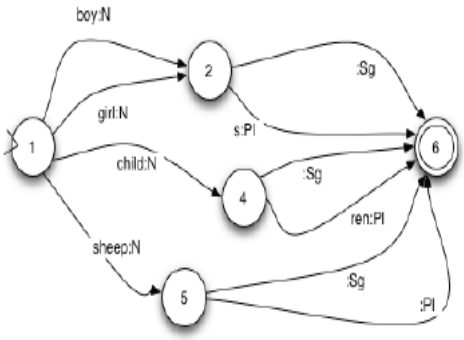

Figure 7: A simple transducer for some English nouns

2. A set of morphophonological, or rather morphographemical rules, in order to be able to generalise over regular suffix alternations

3. An ordered list of closed parts of speech and of exceptional members of the open parts of speech

4. A good reference grammar, from which it is possible to see the regular and the irregular patterns.

2.3 Information retrieval

Good information retrieval from an indexed digital archive rests upon two foundations: a morphological analysator, and a semantic analysis of the vocabulary of the word in question. The morphological analysator is needed in order to group the word forms under the correct lexemes. When looking for Icelandic *fjell* we want hits also for *Ritskrá um fjöll*, thus our analyser must be able to link the wordform *fjöll* to the lexeme **fjell**. For English, this is done with a process called *stemming*. The procedure is as crude as it sounds: The final letter of an unknown wordform is removed, until we get a known wordform. The handful of exceptions *man - men, brother - brethren* is taken care of with a list. For most other languages, the morphology is too complex to revert to such simple devices.

The search will also be more precise if the search clause can be disambiguated. The Norwegian word-form *lodde* is for example ambiguous, it can denote a fish (lkasdjhbioegvh), or it can mean 'to weld'. If the search is given in form of a clause, then a grammatical component can decide whether the noun or the verb is intended, and restrict the search accordingly.

We face another challenge when we encounter real homonymy, such as between English river bank and financial bank. Here, the search machine must first be aware of the homonymy in the first place, and thereafter it must be able to find out which one the user has in mind. The first task is being undertaken a.o. in the work within the WordNet project.

In order to have good information retrieval, we also need a variety of semantic resources. The computer must know the difference between cases such as the the river *bank* and the finance institution, and it must distinguish between the different instances of *Roma* (city, Italy's government, football team, ...). Nouns, bot common nouns and proper names, should receive a semantic classification, so that the analyser can know the difference between animate and non-animate nouns, and between names of persons and names of cars. Such resources are already part of commercial products, and their presence will be felt in the near future. When they are included in common search engines people will see the effect of having a search engine that "understands what you mean"

2.4 Multilingual language technology

Most language technology applications are made for one language only. On the other hand, the one language technology that the customer is actually willing to pay for, is machine translation.

2.4.1 Intelligent dictionaries

By intelligent dictionaries, I mean dictionaries that are able to analyse its input. Thus, if I read an Icelandic internet site and see the clause *lksdfjng hilerjhbl erwjg hv*, I would like to be able to put the cursor on the wordform *ksjdfng*, and get the answer *lksunlkweurhb*. Such dictionaries are already available, e.g. between English and German, but not between e.g. Norwegian and Icelandic, or between Greenlandic and Danish.

2.4.2 Terminology management

By terminology management I mean the requirement, for a multilingual administration, or company, to ensure that the same term is always translated in the same way in all the relevant documents. If, for example, the texts shall have legal status, the choice of word becomes important.

A sentence-aligned parallel corpus of the translated texts, combined with a grammatical analysis of both text versions, will keep track of the terminology usage, and ensure that every time term A is used in the source language, a corresponding term B is used in the target language.

2.4.3 Machine translation

Language technology started out right after World War II, with bright promises that within few years, we would have working machine translation systems. In a famous report in 1968, the ARPANET report, it was stated that all work poured into the field during the 20 years period was in vain.

Today, the situation is far better. Between the commercially interesting languages, there are several machine-translating systems available, and especially for closed domains (texts where we know the topic) the results are quite good.

The Nordic countries now spend vast public resources on machine translation research. Unfortunately, all the effort is directed towards building systems that translate from the national languages into English. As I see it, this means that the Nordic public administration is being opened for English, so that it becomes possible to live in the Nordic countries without knowing the national languages. In order to strengthen the national languages, we need machine translation systems in the opposite direction, so that we can get our children DVDs, our computer game menus, and our EU information in our own languages.

For the minority languages the situation is even more critical. In Finland, the bilingual Finnish-Swedish public sector will be upheld by the means of future machine translation between Finnish and Swedish. When the politicians then notice that the translation costs to Sámi are tenfold, as compared to the translation costs to Swedish, the desire to provide public information in Sámi may be reconsidered and even abandoned.

Making full-fledged machine translation systems is a demanding task, but making translation-assisting software is more manageable. It can be made on the top of an intelligent dictionary and term database, and it will speed up the translation work.

2.5 Speech technology

Making good speech-to-text systems is an ambiguous project. It is worth noting, however, that text-to-speech systems are not that hard to make. The reason for this is that one of the two key components in text-to-speech systems is an automaton that translates the ordinary orthography to some tailored phonetic alphabet. The task of making such an automaton is like formalising the chapter on pronunciation rules in your favourite reference grammar (only this time, all the

exceptions must be listed, and not only shortly exemplified). If the resulting string is connected to a speech system for another language, than you get a machine that speaks your language, but in a foreign accent.

3 Some case studies

I will have a look first at localisation and then at grammatical transducers. The main focus will be on work for Sámi, but I will contrast Sámi with Yoruba wrt. localisation, and Sámi with Greenlandic wrt. grammatical transducers.

3.1 Localisation

3.1.1 The Sámi languages

There are 6 Sámi written languages. The largest is Northern Sámi, with perhaps 18000 speakers, whereas the remaining 5 languages have less than thousand speakers. Here I will give you a short overview over the status quo of Sámi language technology, and of what it has taken to bring us there.

Sámi There are 6 Sámi written languages, as seen in the overview in table 1.

Lg	Speakers	Alph	extra letters
Southern	< 500	latin	1
Lule	< 1000	latin	2
Northern	16600	latin	7
Inari	< 300	latin	4
Skolt	< 300	latin	12
Kildin	< 500	cyrillic	13

Table 1: Sámi literary languages

As an illustration, the alphabets of Skolt and Kildin Sámi are given in figures 8 and 9, respectively

Alphabet:
A a, Â â, B b, C c, Č č, Ʒ ʒ, Ǯ ǯ, D d, Đ , E e, F f, G g, Ǧ ǧ,
G̨ g̨, H h, I i, J j, K k, Ǩ ǩ, L l, M m, N n, Ŋ ŋ, O o, Õ õ, P p,
[Q q], R r, S s, Š š, T t, U u, V v, [W w], [X x], [Y y], Z z,
Ž ž, Å å, Ä ä, [Ö ö], ´

Figure 8: Skolt Sámi alphabet

Alphabet:
А а (Ӑ ӑ), Ӓ ӓ (Ӑ̈ ӑ̈), Б б, В в, Г г, Д д, Е е (Ӗ ӗ), Ё ё (Ӗ̈ ӗ̈),
Ж ж, З з, И и (Ӣ ӣ), Й й, Ӈ ӈ, К к, Л л, Ԯ ԯ, М м, Ӎ ӎ, Н н, Ң ң,
Ӊ ӊ, О о (Ӧ ӧ), П п, Р р, Ԓ ԓ, С с, Т т, У у (Ӯ ӯ), Ф ф, Х х, Ц ц,
Ч ч, Ш ш, Щ щ, Ъ ъ, Ы ы, Ь ь, Ѣ ѣ, Э э (Ӭ ӭ), Ӧ ӧ (Ӧ̈ ӧ̈), Ю ю
(Ю̄ ю̄), Я я (Я̄ я̄), J j, Ѣ ѣ, ’

Figure 9: Kildin Sámi alphabet

3.1.2 Sámi Localisation – a success story

The biggest achievement for Sámi localisation is the fact that Northern Sámi keyboard layout is included, out of the box, in three different national settings, no matter where you buy your computer, from the OS versions Linux KDE 3.0, Mac OS 10.3, Win XP SP2 and higher. This means that we may safely assume that every user will be able to type Sámi words, as soon as the relevant keyboard is chosen. This achievement is a result of a decade of hard work, including grass-root conferences among language users in order to arrive at a consensus among earlier competing layouts, as well as standardisation work in the relevant standardisation bodies (one of them, CEN TC304 had its secretariat here in Reykjavik), lobbying work and explicit pressure from our state administrations upon the OS vendors, and volunteer work within the Linux movement. The result is that basic Sámi localisation is supported by all the OS's, whereas languages with more than thousand times as many speakers are not.

There are keyboards and graphical user interfaces for some languages in the major operative systems already. Table 2 gives an overview of the situation in april 2005 (the dash indicates that the figures were not available to the author).

OS	keyboard	GUI
Windows XP	51	33
Mac OS X	42	-
Linux KDE	-	88

Table 2: Keyboard layouts and graphical user interface for the major operative systems

We may have a closer look at the largest languages *without* localisation support, and the smallest languages *with* support, in table 3 (language rank and number of speakers are from the Ethnologue).

12 largest lgs with limited support				12 smallest lgs with basic support or more			
Rank	Speakers	Name	Country	Rank	Speakers	Name	Country
26	41.0	Bhojpuri	India	2108	0.014	Inuktitut	Canada
33	30.0	Siraki	Pakistan	1971	0.017	North. Sámi	Nordic
35	24.0	Maithili	India	1752	0.022	Cherokee	USA
37	23.0	Oriya	India	1344	0.047	Greenlandic	Greenland
39	22.0	Burmese	Myanmar	1343	0.047	Faroese	Denmark
40	22.0	Hausa	Nigeria	1304	0.050	Maori	NZ
44	20.3	Awadhi	India	991	0.940	Gaelic	Scotland
47	20.0	Yoruba	Nigeria	601	0.250	Icelandic	Iceland
51	17.0	Sindhi	Pakistan	517	0.330	Maltese	Malta
53	16.0	Nepali	Nepal	407	0.500	Breton	France
55	15.0	Amharic	Ethiopia	370	0.580	Welsh	UK
59	13.7	Assamese	India	292	0.910	Basque	Spain
60	13.0	Haryanvi	India	130	4.000	Georgian	Georgia

Table 3: The 13 largest languages without – and the 13 smallest languages with – basic localisation support in at least one of the major operating systems

The largest languages with marginal or no IT support (approximatively 6400 lgs), are typically African languages or Indian languages other than the 22 Indian official state lgs, or they are languages without official status in an independent country, especially in former British and French colonies. Languages with IT support, on the other hand (the remaining 100 lgs), are languages with official status in an independent country without a colonial past, and rich and monolingual speakers, they are (most of) the official state languages of India, or they are minority languages

with a strong government backing them up (Western Europe, Canada, New Zealand). It is this latter group we see represented at the right hand side of table 3.

3.1.3 Future perspectives for massive multilingual localisation

Table 3 will very soon be outdated, as there is much work going on within localisation around the world, especially in Africa. Minority language activists typically turn to the open source movement, where they can localise whatever language they want. Microsoft and Apple are more restrictive, and only localise when they see a reason for it. Looking for a moment at the desktop war, and focusing upon Microsoft vs. Linux, I think it can be characterised as in table **??**.

Microsoft...	Linux...
has a dominant market pos	comes for free
has more 3-party software providers	does not crash
has better plug-and-play	is open source
has far better language technology (spell checkers & grammar checkers)	**can be localised to any lg with speakers who care to do the job**

Table 4: The Microsoft vs. Linux desktop war

Both parties will probably try to match the competitor. I expect Microsoft to want to extend both the localisation and the spell checkers to more languages (reacting to critique from the open source community). I also expect them to choose a statistically based approach to the spell checkers, although they will chose grammatically-based ones when they are readily available (as for the Nordic languages).

I expect the Linux community to take the lg tech challenge more seriously. The problem with this is that language technology cannot be done "the Linux way" , as global volunteer hacking projects. Language technology projects need teams of several programmers, lexicographers, philologists and computational and theoretical linguists, working together for years. At the moment, only Microsoft has both the resources and the will to finance such projects. Government-funded language research institutions certainly have such resources, but at least so far, they have been reluctant to pay for the development of open-source products. The typical situation is then that the basic work is being done by public funding (making of dictionaries and reference grammars), and then Microsoft comes in, buys the right to use the resources, and pays the final 2-5 % it costs to turn these resources into authoring tool products.

On the other hand, we see that more basic tools become open source. Examples include the Stuttgart *sfst* finite transducer, and the Odense *vislcg* disambiguator. Also, publically funded dictionary projects now start to make their lexica accessible to open source projects. In order to see authoring tools on all desktops, this tendency should be strengthened.

The point here is not to boast of how lucky Sámi users are, but to show this as an example of how to proceed. A language for which this issue is being hotly debated at the moment is Yoruba. With 20 million speakers, Yoruba is far from a minority language. When it comes to keyboard layout, there is no consensus, and many even hold the position that Yoruba should sacrifice its orthography due to the computer problems. This last position I will simply dismiss as being erroneous. After the establishment of the Unicode standard, the only factor blocking users from implementing their orthographies as they should be, is ignorance.

3.1.4 How were the Sámi results achieved?

After an early period of ad hoc solutions, differing from operating system to operating system and from country to country, at the wake of e-mail and the internet in the mid-nineties, the Sámis found themselves hampered by a large number of conflicting standards, both with regard to character encoding and to keyboard layouts. In order to meet this challenge, the Norwegian

Parliament in 1996 appointed a committee for computer standardisation[1]. The work on character set standardisation was only intended as an interlude before the arrival of Unicode, and it will not be reported here, instead we will focus upon the work on keyboard standardisation.

Put in a list fashion, the work may be summarised as follows.

There were many keyboard layouts available We collected all of them, and compared them to each other. Letters that had the same positions in all former keyboards kept their positions

The keyboards had different major-language keyboards as a starting point We decided to make the keyboards as similar as possible to the majority-language keyboards in the country where they were used. Thus, Finnish Sámis got a keyboard based upon the Finnish keyboard, Swedish Sámis one based upon the Swedish keyboard, etc. Thus, the users could find symbols like the @, the §, etc. in the positions they would assume them to be placed.

The keyboards had used different keys for the 7 Sámi letters We had to decide which keys to use for Sámi letters, and which keys to keep untouched. As part of a conservative strategy, we kept the possibility of writing Sámi and the majority language with the same keyboard (æ, ø, å were kept), thereby facilitating typing place- and person names, whereas the non-Nordic letters q, w, x were sacrificed)

We had to decide how to place the Sámi letters We investigated the text frequency of the relevant letters in a large corpus of Sámi texts. The most common letters were given more prominent positions.

The replaced letters had to be put somewhere As a rule, we put the replaced letters one level up. So, when the key **W** gives *š*, then, in order to get *w*, you press **option-w**, etc..

Keyboard layouts should be tested by skilled typists For Northern Sámi, we were not able to carry out systematic testing, this clearly was a weakness of the work.

The (pre-Unicode) version of the Northern Sámi keyboard for Macintosh in Norway, arrived at as aresult of this work, is shown in figure **??**.

The Skolt Sámi keyboard was a greater challenge, as there were more letters to include. Here, the standardisation work was easier, as there was already a layout for MS-DOS available, and no competing layouts. The standardisation work was then reduced to rearrange the symbols that had been replaced, and to include the possibility of typing other Sámi letters. Cf. figure **??**.

When a language is written in different countries, such as Fulfulde in both former French and English colonies, one uniform keyboard layout is not a realistic goal. Placing the Fulfulde letters on the same positions in Anglo-Fulfulde and Franco-Fulfulde keyboards, should get high priority, but after that, the different ex-colonies should pay attention to different official languages.

If it turns out that consensus is impossible (Spanish is for example represented with two different keyboard layouts), one should bear in mind that it is no catastrophe. How you type in your text is irrelevant to me as long as I can read the result, i.e. as long as you encode your letters in a way that I can recognise. The bonus of a unified keyboard standard is that it makes it easier to teach typing skills, and to change from one computer environment to another.

A further key localisation application is a sorting algorithm. Language communities have different conventions for sorting their letters, and the computer should be made aware of them. In addition to sorting the main letters of the indigenous alphabet, a sorting algorithm should have a fallback for treating non-native letters. The sorting algorithm made for Northern Sámi fulfil these criteria [2], this (or indeed any) sorting algorithm has only been installed on the Linux platform, though, although there is work underway to implement it on Windows as well.

[1] The committe members were Audun Lona (leader), Inger Marie Gaup Eira, Edmund Grønmo, Hannu Kangasniemi, Peter Sarri and Trond Trosterud.

[2] Cf. the sorting string at *http://www.hum.uit.no/a/trond/sorting.html*

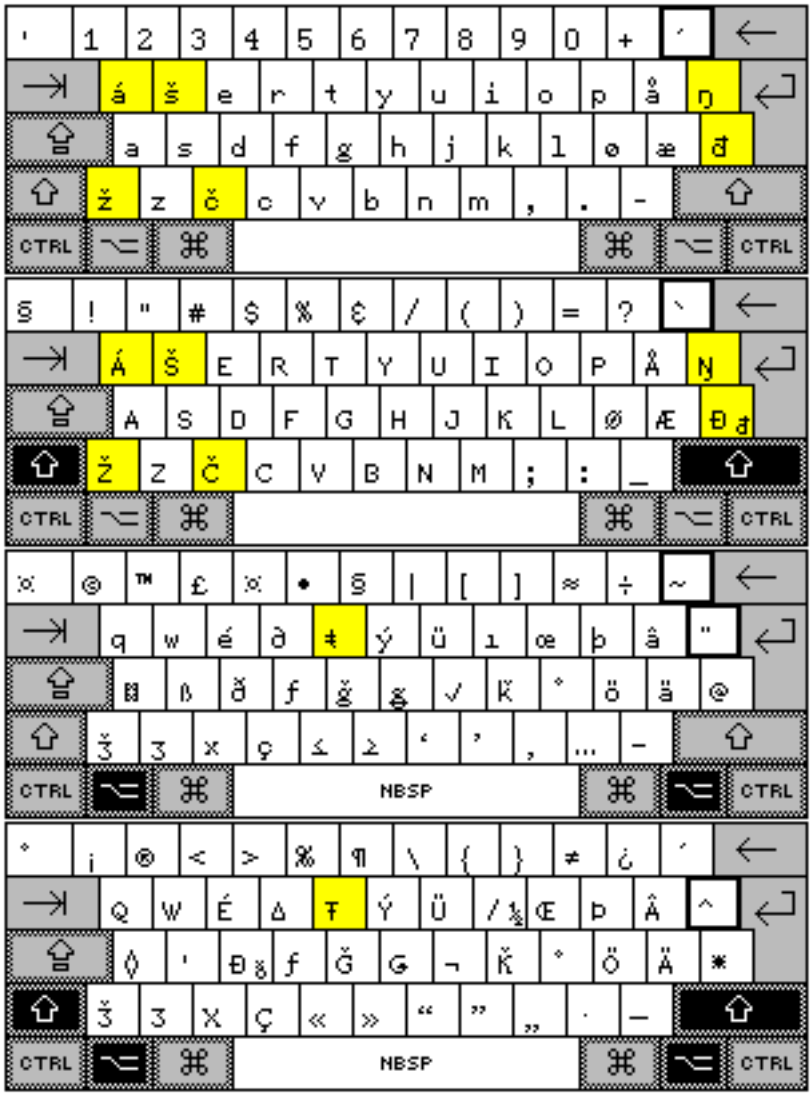

Figure 10: Northern Sámi keyboard for Macintosh, Norway

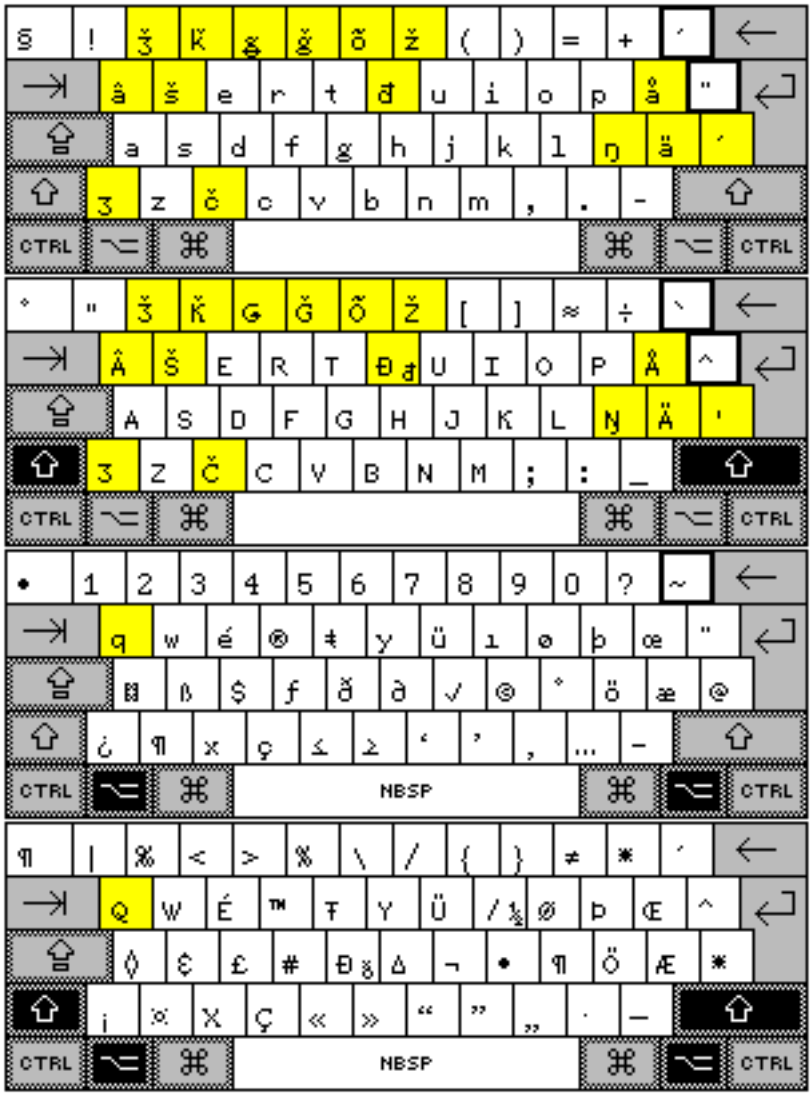

Figure 11: Skolt Sámi keyboard for Macintosh

3.2 Yoruba localisation

As an illustration of a language that has not reached the consensus described above, let us have a look at Yoruba. Yoruba has 19.3 million speakers, and it has official status in Nigeria. It is a tone language, and tone is marked in the latin-based orthography. As a result of this, it has many letters not found in the a-z alphabet, namely á, à, é, è, ẹ, ẹ́, ẹ̀, í, ì, ó, ò, ọ, ọ́, ọ̀, ṣ, ú, ù .

Yoruba has no official support on any OS, but there is work in progress for making the Linux KDE platform available in Yoruba. At the moment, there is a discussion on removing the diacritics from the orthography, as they cause so much trouble for the use of Yoruba in computers.

My view on that issue is that the computer should adjust to humans, and not vice versa. Orthographies and keyboard layouts should be designed according to linguistic and ergonomic principles. We linguists invented these diacritic signs, and we should engage in finding solutions.

It should also be mentioned that several Yoruba speakers are optimistic with respect to the situation, e.g. Samuel Olamijulo, here quoted from a letter to the A12n-forum on localisation for African languages, where he points at one of the many keyboard layouts available for Yoruba.

> Typing ALL Yoruba Letters, undermarks and tonal signs included, is now easy with practice, using free Arial Unicode MS font and ABD Yoruba Keyboard . Yoruba Desktop Publishing is making tremendous progress with inputs from many good contributors all over the world. One important persisting user headache is in communication accross e-mail , yahoogroups, other web forums, websites and other Internet applications even when Arial Unicode MS or other Unicode Compatible fonts are used in the creation of the message. Arial Unicode MS font in MS Word 2003 and ABD Yoruba Keyboard are available for free.

As can be seen from the quote, Yoruba now finds itself in the same position as Sámi faced back in 1995: There are several layouts available, there is work going on in different places, but there is no consensus, and indeed no official support.

3.3 Northern Sámi language technology

We now turn to Northern Sámi language technology. There is work going on within the following areas:

- Basic tools: Parser and disambiguator
- Spell checker
- Pedagogical programs
- Terminological database
- Encoded mono- and bilingual corpora

3.3.1 Parser and disambiguator

Northern Sámi is an Eurasian Turkish-type language (adverbial cases, morphological suffixation, no declension classes), but with heavy influence from the neighbouring Germanic languages, such as non-segmental inflectional processes such as stem-internal diphthong and consonant alternation. Each lexeme can have several tens of inflected forms, verbs and adjectives have over 100 inflected forms. The stem modulations make stemming inappropriate as a method for information retrieval.

Let us have a look at the sentence *Áhčči lea oastán munnje divrras sabehiid* 'Father has bought me an expensive pair of skis'. Analysed by the morphological parser being made at the University of Tromsø (*http://giellatekno.uit.no*), it comes out as seen in figure m-alle.

Most of these morphological analyses are wrong in this particular context. Our disambiguator removes the inappropriate ones (and adds grammatical function), as can be seen in figure **??**.

```
"<Áhčči>"
        "áhčči" N Sg Nom
"<lea>"
        "leat" V Ind Prs Sg3
"<oastán>"
        "oastit" V PrfPrc
        "oastit" V* N Actor Sg Nom PxSg1
        "oastit" V* N Actor Sg Gen PxSg1
        "oastit" V* N Actor Sg Acc PxSg1
        "oasti" N Sg Nom PxSg1
        "oasti" N Sg Gen PxSg1
        "oasti" N Sg Acc PxSg1
"<munnje>"
        "mun" Pron Pers Sg1 Ill
"<divrras>"
        "divrras" A Attr
        "divrras" A Sg Nom
"<sabehiid>"
        "sabet" N Pl Gen
        "sabet" N Pl Acc
"<.>"
        "." CLB
```

Figure 12: Morphological analysis of the Northern Sámi sentence *Áhčči lea oastán munnje divrras sabehiid* 'Father has bought me an expensive pair of skis'

```
"<Áhčči>"
        "áhčči" N Sg Nom @SUBJ
"<lea>"
        "leat" V Ind Prs Sg3 @+FAUXV
"<oastán>"
        "oastit" V PrfPrc @-FMAINV
"<munnje>"
        "mun" Pron Pers Sg1 Ill @ADVL
"<divrras>"
         "divrras" A Attr @AN>
"<sabehiid>"
        "sabet" N Pl Acc @OBJ
"<.>"
```

Figure 13: Disambiguated version of the Northern Sámi sentence *Áhčči lea oastán munnje divrras sabehiid* 'Father has bought me an expensive pair of skis'

These tools are still under construction, but test results show that the program is able to disambiguate unknown text with a precision of 94% (morphology) and 93% (syntax), with a recall of 99%, and an accuracy of 1.056.

3.3.2 Applications

The parser and disambiguator for Sámi are being put to use in several applications:

Spell checker The sámi parliament has issued a 3-year project geared towards making a spell-checker for two of the Sámi languages. Excluding the time spent on the basic parser technology, the spell-checker is estimated to take 14 man-years, including work on the Sámi standard.

Hyphenator We will have a morphologically based hyphenator, that incorporates a morphological analyis done by the Sámi analyser (in order to find the word boundary), with a set of phonotactic rules in order to find possible hyphen insertion points. In cases of conflict, the word boundary wins.

Pedagogical programs Utilising work done at the University of Southern Denmark, we are using the parser as a basis for making interactive pedagogical programs.

The input format to the pedagogical programs, for the same sentence as was shown above, cah be seen in figure **??**.

```
S:n('áhčči',sg,nom) Áhčči
P:g
=D:v('leat',ind,pr,3sg) lea
=H:v('oastit',pcp2) oastán
A:pron('mun',<pers>,1sg,ill) munnje
Od:g
=D:adj('divrras',attr)  divrras
=H:n('sabet',pl,acc) sabehiid
```

Figure 14: The Northern Sámi sentence translated into underlying ped-format

The graphical outcome of the analysis is shown in figure **??**. The sentence may also be displayed, and interactively analysed, with a set of 200 other Sámi sentences, at *http://beta.visl.sdu.dk/visl/smi/*.

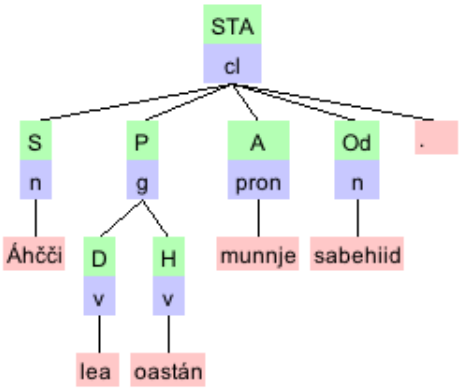

Figure 15: The visl graphical version of the Northern Sámi sentence

3.3.3 Future plans

We plan to port our solutions for Northern Sámi to all the other Såmi languages as well, and eventually to other Uralic languages.

When conducting language technology projects, large part of the planning costs go into setting up an infrastructure. Commercial companies naturally keep this infrastructure to themselves, as this is part of their competitive advantage, as compared to newcomers in the field. In Tromsø, we plan to publish our infrastructure as part of an open-source how-to for language technology projects. Even though the core tools for our morphological parsers are not open-source (*twolc, lexc, xfst* are part of the Xerox toolbox, see *http://www.fsmbook.com*, they are freely available for non-profit project. People wanting a full open-source platform may turn to alternatives, such as the Stuttgart finite state tool *sfst* (*http://www.ims.uni-stuttgart.de/projekte/gramotron/SOFTWARE/SFST.html*).

3.4 Greenlandic

The most obvious language to build a morphological automaton for is Greenlandic, a polysynthetic language that combines an average morphological structure (7 cases, 6 persons, 2 numbers) with an extensive number of derivational affixes.

At present, a list-based spell-checker for Greenlandic is under construction. So far, the results have not been too encouraging. Even with an exceptionally high number of wordforms (350000), the spell-checker only recognises approximately 40 % of the words in running text, i.e., it has a precision of 0.4, which is so bad as to leave it totally useless.

[**?**] were able to make a good a grammatically-based transducer and disambiguator for French in a couple of man-months. Making a transducer for Greenlandic would very return better results than the depressing 40 % of the list-based spell checker.

4 Co-ordinating documentation and language technology

There is no doubt minority languages will need language technology in order to function as literary languages. Endangered languages, typically without a written standard, and with few speakers left, are in a totally different situation. What are the perspectives for co-ordinating documentation, such as writing reference grammars and collecting corpora for endangered languages?

For university lingustics, languages with few speakers are as interesting as languages with many speakers, as the grammatical intricacies are the same. Even more so: Languages where the linguist in question may be a pioneer, or where his or her work will necessarily have lasting impact, may even be more attractive than languages with an abundance of both speakers and practising linguists. On the other hand side, the increasing commercialisation of academia may change this, forcing linguists to work on commercially more interesting languages.

When languages are about to vanish, we want basic documentation:

It is not obvious that resources should be geared towards making transducers etc. but lexicographical work should be conducted in a structured way if large corpora are available, they could be annotated by a parser. A parser is also a handy tool for checking the validity of the rules of the reference grammar.

5 Conclusion

Language technology solutions is both a *sine qua non* for minority language communities wanting to use their language for administrative and literary purposes, and a necessary tool for reference work. Linguists, programmers and language activists should co-operate on making these tools.

The work is time-consuming, but apart from the cost of computers to run your Linux distro on (and in some cases scanners), the work on text-based language technology does not require expensive equipment. Thge amount of work done will thus be dependent upon the possibilities of finding people being willing to do it.

I am optimistic on behalf of language technology for at least a noticeable part of the 6500 languages of the world. But I would like to see more of my colleague linguists to join in.

References

[CT95] Jean-Pierre Chanod and Pasi Tapanainen. Tagging french - comparing a statistical and a constraint-based method, 1995.

[ONM00] Kemal Oflazer, Sergei Nirenburg, and Marjorie McShane. Bootstrapping morphological analyzers by combining human elicitation and machine learning. *Computer Engineering Technical Report*, BU-CE-0003, 2000.