Deduction of Numeral Grammars

Harald Hammarström
Chalmers University
harald2@cs.chalmers.se

Abstract.

We describe an unsupervised method to learn how numeral expressions (e.g fifty-three, ninety-three etc) are formed. This problem is seen as finding a minimal-size set of atomic forms and concatenation rules, and the method for abstracting powerful concatenation rules is through k-means clustering of the strings on the edit-distance. We evaluate the method on the number words for 1–101 in 90 diverse languages and discuss success rates and limitations.

1 Introduction

In this paper we shall describe experiments on automatic abstraction of natural language numeral systems. To be more precise, for a given list of the names for the numbers 1 to n in a language we get a grammar and a set of atoms. By grammar we simply mean a set of concatenation rules and atoms are those items which show no possibilities of regular decomposition. For the problem to be interesting, we also want the grammar to be as tight and complete as possible. For example, in English the numerals ≤ 100 are as follows:

one	eleven	twenty-one	thirty-one	...	seventy-one	eighty-one	ninety-one
two	twelve	twenty-two	thirty-two	...	seventy-two	eighty-two	ninety-two
three	thirteen	twenty-three	thirty-three	...	seventy-three	eighty-three	ninety-three
...	...	...	...	...	...	...	...
ten	twenty	thirty	forty	...	eighty	ninety	a hundred

The atoms would be 1–12 + variants and the grammar would consist of rules like:

```
{thir|four|fif|six|seven|eigh|nine}teen
{twen|thir|four|fif|six|seven|eigh|nine}{ty-one|ty-two|...|ty-nine}
...
```


Proceedings of the Ninth ESSLLI Student Session
Laura Alonso i Alemany and Paul Égré (editors)

It's obvious that such grammars are subsumed by context-free grammars where the atoms correspond to terminals, and since the sets of strings discussed in this paper are all finite, regular expressions can account for them too.

The grammar can be non-empty whenever there are regularities like in the English numeral system. The methods used aren't specialized for numerals but work on any set of strings that has similar regular properties. However, numeral systems typically have strong and abundant rule-like features, yet at the same time irregularities and idiosyncrasies so characteristic of natural language. This makes them ideal for testing one's methods and moreover, numeral systems are much more comparable *across* languages than almost any other subset of grammar.

We have done experiments for numbers 1–101 on about 90 languages from a numerals project centered around GF(Ranta 2004) at Chalmers University. The methods and results are discussed below.

2 Method

The problem can be formulated as picking out the optimal grammar from the giant set of combinatorially possible grammars. Instead of brute-force going through the whole search space we can heuristically deduce what at least a near-optimal grammar should be. This (tractable) computation is carried out along the following lines:

- Iterate until no more rules can be abstracted:
  - Generate a k-clustering for $k = 1, 2, 3, .., i$ (for some suitable i)
  - Select the optimal clustering on basis of average rule size
  - Rule abstraction from clusters

Strings that have been abstracted into a rule in an earlier iteration can still be subject to further abstraction.

2.1 Clustering

The clustering phase uses a straightforward k-means classifier(Russell and Norvig 2003). The similarity measure is the edit-distance (Wagner and Fischer 1974) with settings that penalize different length strings more than same-length different strings. Some quick checks indicated that the most useful edit costs were 0.25 for substitution versus 1 for deletion.

All the strings are in our internal 8-bit ascii representation which is that of the language itself or a phonematic approximation of the orthography of the source. It is truer to language to use some kind of phonemic distance (rather than orthographic distance) as in e.g (Kondrak 2002) but that would be overkill for this experiment.

Now, the crucial question is how many clusters one should divide into because, as shall be explained, the rule abstraction is done on basis of the cluster divisions. Table 9.1 has some examples of clustering outcomes for some k:s over English.

The classic clustering dilemma is to find "the right" k. Ideally one will want that the clusters are as tight and as disjoint as possible, but there's no simple way to optimize for maximum tightness, disjointness or both for *different* k:s that does not trivially favour the case where each

k	x	n	w	k	x	n	w	k	x	n	w
1	0	0	0.0	26	73	11	58.4	51	74	14	44.3
2	0	0	0.0	27	63	8	69.4	52	71	15	42.6
3	0	0	0.0	28	56	9	50.9	53	83	16	43.8
4	7	1	35.0	29	59	9	57.2	54	80	17	42.9
5	7	1	35.0	30	81	11	66.1	55	75	12	51.9
6	0	0	0.0	31	80	13	51.8	56	81	16	48.2
7	0	0	0.0	32	71	13	45.9	57	72	12	54.8
8	7	1	35.0	33	84	14	51.4	58	83	17	44.2
9	7	1	35.0	34	58	10	56.5	59	84	18	42.7
10	7	1	35.0	35	87	13	60.7	60	74	20	34.2
11	23	3	67.7	36	81	13	55.8	61	80	13	57.8
12	15	2	61.5	37	79	14	51.5	62	81	16	45.8
13	25	4	57.3	38	80	12	60.5	63	73	16	40.3
14	27	4	60.8	39	78	13	50.6	64	72	15	45.8
15	23	3	46.3	40	87	13	60.2	65	65	15	39.9
16	39	5	74.2	41	83	16	47.1	66	65	15	38.9
17	55	7	75.9	42	82	15	49.5	67	65	15	41.3
18	47	6	65.8	43	81	16	45.9	68	70	15	44.1
19	63	8	64.4	44	87	16	49.9	69	78	18	40.9
20	63	8	70.4	45	83	16	46.9	70	65	15	39.9
21	55	8	66.6	46	75	15	45.2	71	50	12	34.6
22	63	8	69.4	47	87	15	52.9	72	65	14	44.9
23	63	9	60.1	48	75	18	38.4	73	43	9	40.3
24	66	11	53.5	49	84	18	42.8	74	25	7	38.4
25	66	11	54.6	50	87	17	46.9	75	79	13	54.9

Table 9.1: x is the number of strings affected in rules induced by the clustering. n is the number of induced rules and w their average size. There is a random factor in initializing but the result vary only meagerly from run to run.

object is a single cluster. Although there are some interesting general approaches to this problem around (Ghosh 2003), they require commitment to supervision or models that we might not want to require on our sets of strings.

So instead we use the average induced **rule size** as the optimization goal. By rule size we mean how many cases a rule applies to plus the length of the generalization. So e.g a rule `{thir|four|fif|six|seven|eigh|nine}teen` gets size 7 * 4 = 28. This is unsupervised and entertains the intented intuition that we want strong rules in the end.

From the average rule sizes in table 9.1 we see that the best cluster sizes for English seem to be around 10 or 20, reflecting the fact that English is base-10 and we get general rules if we cluster into teens, twenties, thirties, ..., nineties plus one cluster of units or ten different clusters as in one for each unit (for they have nothing in common except not belonging to the other classes).

2.2 Rule Abstraction

For each cluster, the best generalization is calculated by matching all pairs of cluster members and greedily choosing the longest most applicable one. Most applicable means that the highest number of members of the cluster satisfy it. A generalization of a set of strings is a common prefix, suffix or circumfix (other generalizations, such as infixes (Yu 2003), are possible in natural language but did not seem to occur in our sample). For example, if a cluster contains `{twenty, thirty, forty, fifty, sixty, seventy, eighty, ninety, ninety-four}`, the most applicable generalization is a suffix `-ty` which is common to all members except one (= 8 which is more than e.g 2 for `-rty`, `f-ty`, `ninety-`) and it is longer than `-y`, which would also apply to the same eight members.

To filter out a large class of unwanted rules we also postulate that a rule must:

- be built on more than one example
- generalize more than one character

The last criterion is because a lot of one char suffixes and prefixes occur by chance.

A few more comments are in order:

- Once a rule has been extracted from a cluster, any other cases outside the cluster that match the rule are incorporated.
- All rules are disjoint in the sense that the same string is never part of two parallel rules.
- Maximally only one rule is abstracted from a cluster at each step, since rules are never revoked and thus it's very important that the rules are sound and maximal.

2.3 Related Methods

The method described is similar to induction algorithms by de Marcken (de Marcken 1996) and (Baroni 2000) in that it sees rules as compression – a good rule is one which makes description shorter. And like Goldsmith's Linguistica (Goldsmith 2001) it relies on the existence of affixes that are frequent, as food for generalizing. The difference in our case is the use of clustering, where potentially variants like `three` and `thir` have a better chance of being recognized. It is a turnoff,

then, to note that in the experiments aimed at in this paper we have not set out to exploit this property. It is also beyond the scope of this paper to look empirically at how clustering predicts affix divisions compared to Goldsmith's heuristic or Johnson and Martin's (Johnson and Martin 2003) hub detection.

3 Experiments

3.1 Evaluation Metric

Each language has been annotated with a gold standard of the components of its numeral system. The components are the atoms and their allomorphs, other morphemes and orthographic links (e.g **and**, **-**, spaces etc), "irregular extensions", and formation rules. Irregular extensions are expressions that are not monomorphemic but occur in only one or two cases – too few to merit a rule. To give the reader an idea, the raw annotation for English is given below:

```
#English
b = 10

a = {}
a[1] = ‘‘one’’
a[2] = ‘‘two’’
a[3] = ‘‘three’’
a[4] = ‘‘four’’
a[5] = ‘‘five’’
a[6] = ‘‘six’’
a[7] = ‘‘seven’’
a[8] = ‘‘eight’’
a[9] = ‘‘nine’’
a[10] = ‘‘ten’’
a[100] = ‘‘hundred’’
a[1000] = ‘‘thousand’’

irr_ext = {}
irr_ext[11] = (1, ‘‘eleven’’, ‘‘leaves’’)
irr_ext[12] = (2, ‘‘twelve’’, ‘‘leaves’’)

v = {}
v[2] = [Variant(‘‘twen’’, {PART_ROLE: ‘‘small’’})]
v[3] = [Variant(‘‘thir’’, {PART_ROLE: ‘‘small’’})]
v[4] = [Variant(‘‘for’’, {VARIANT: ‘‘ten’’, PART_ROLE: ‘‘small’’})]
v[5] = [Variant(‘‘fif’’, {PART_ROLE: ‘‘small’’})]
v[8] = [Variant(‘‘eigh’’, {PART_ROLE: ‘‘small’’})]
v[10] = [Variant(‘‘ty’’, {VARIANT: ‘‘ten’’}),
         Variant(‘‘teen’’, {VARIANT: ‘‘teen’’})]
```

```
AND = Element(‘‘ and ‘‘, AND)

cr = {}
cr[13] = Rule([(MOD, REM, Compound(ADD, [SMALL, BIG], ‘‘teen’’))])
cr[20] = Rule([(MOD, Q, Compound(MUL, [SMALL, BIG], ‘‘ten’’)),
               (RES, REM, Compound(ADD, [BIG, SMALL]))])
cr[100] = Rule([(MOD, Q, Compound(MUL, [SMALL, SPACE, BIG])),
                (RES, REM, Compound(ADD, [BIG, SPACE, AND, SPACE, SMALL]))],
               None,
               OVERRIDE)
cr[1000] = Rule([(MOD, Q, Compound(MUL, [SMALL, SPACE, BIG])),
                 (RES, REM, Compound(ADD, [BIG, SPACE, SMALL]))],
                None,
                OVERRIDE)
```

The reader need not worry about the syntax and semantics of the annotation scheme, but should assume it allows retrieval of the details of each numeral system in the sample in a uniform way. Since the database was originally designed for typological comparison, it happens annoted in too high detail than what is needed, and we shall now proceed to the actual benchmark.

The irregular extensions, atoms and allomorphs of the gold standard are treated as atoms, just as orthographic links and non-numeric morphemes are incorporated into the formation rules. Now, think of a rule as a set of strings, namely those strings it generates when applied to its domain. The numeral system as a whole, a set of atoms and a set of rules, is now a set of strings (of atoms) and set of sets of strings. There's no reason to handle the atoms separately, so a numeral system can be treated as a set of sets of strings.

So, for clarity, denote an induced numeral system $I = \{r_1, \ldots, r_m\}$ vs. the gold standard $G = \{R_1, \ldots, R_n\}$. Of course $\bigcup I = \bigcup G =$ 1–101 in some language.

A rule r is a set of strings, $s_1, \ldots, s_k$, and theoretically these may belong to as many as k different "real" rules according to the gold standard, but (hopefully), they only belong to i different real rules. In these terms, the precision of a rule is $\frac{k-i+1}{k}$. So e.g, if they all belong in one rule in the gold standard too, then the precision is a perfect 1 and vice versa. Formally:

$$precision(r, G) = |\{R_x \in G | \exists s \in r, R_x\}| \tag{9.1}$$

Now, define the accuracy as:

$$Accuracy = \frac{\frac{1}{|I|}\sum_{r \in I} precision(r, G) + \frac{1}{|G|}\sum_{r \in G} precision(r, I)}{2} \tag{9.2}$$

Note that this accuracy-metric only measures which strings belong to which rules, and we leave out details as to whether the actual formation/split in the rule is correct.

Again, we take a fictions example of a fraction of English:

3.2 Results

We give a sum mary in table 9.2 of the algorithm's performance on the 90 languages in the test set. Kwaza (van der Voort 2000) and Kayardild (Evans 1995) have been excluded since they do not really have numerals over five.

I	G	$precision(r_1, G)$	$precision(R_2, I)$
$r_1 = \{one, seventeen\}$	$R_2 = \{one, two, \ldots, ten\}$	$\frac{1}{2}$	$\frac{9}{10}$
$r_2 = \{nineteen\}$	$R_3 = \{thirteen, \ldots, nineteen\}$	1	$\frac{?}{7}$
$r_3 = \{two, \ldots, ten\}$	...	...	...
$r_4 = \ldots$			

We will give examples to illustrate the ratings. For German we get:

```
{{dr|zw}ei|neun|acht|fünf|sechs|ein|vier|sieben} und {{neun|zwan|sieb|fünf|vier|sechs|acht}zig|}
{{dr|zw}ei|acht|drei|fünf|neun|sechs|sieb|vier|} und dreissig
{|{dr|zw}ei|acht|drei|fünf|neun|sechs|sieb|vier|}zehn
hundert{|eins}
```

And the atoms `eins, ..., neun, dreissig, elf, zwölf` which is pretty much optimal. One should think about whether dressig should be a separate case or part of a larger `-ig` rule, but the algorithm favoured `-zig` abstraction (7*3 vs. 8*2). The situation is not uncommon, e.g many Romance languages have 30 with a suffix slightly different from 40, ..., 90 as in `trenta, quaranta, cinquanta, seixanta, setanta, vuitanta, noranta` (in Catalan (Maria Mas and Vergés 2000)).

The German case illustrates another prominent feature, namely grouping of words like `zwei` and `drei`, that are short and have a common sequence of length 2. This is responsible for unfortunate abstractions in e.g Guahibo (Queixalos 1986), Kabardian (Colarusso 1992), Malay (Dodds 1977). Classical Arabic (Haywood and Nahmad 1962) and Pashto (Penzl 1955) are even more vulnerable as they are implemented in romanizations of a short-vowel-less Arabic script, and so generally have shorter string and chance resemblances become harder to distinguish. These unwanted generalizations could easily be made illegal by imposing restrictions also on length 2 abstractions. But we don't want to do this on the general level since length 2 abstractions are also often wanted and unproblematic (e.g Biblical Hebrew (Nyberg 1972), also short-vowel-less, abstracts fine).

The other insuperable problem is tone-variating Tibetan (Herbert Bruce 1978) or highly inflecting languages like Polish and Old Church Slavonic (Nandriş 1959). Similarly, the closely related sandhi-intensive Dravidian languages Irula (Perialwar 1978), Kodagu (Balakrishnan 1977) and Tamil yield many classes of abstractions with three or four members. We see no simple remedy for treating these cases in general.

There are many languages that use base-20 in addition to a base 5 or 10, especially for numbers ≤ 100. E.g Nootka (Folan 1986), Breton (Trépos 1980), Maybrat (Philomena Hedwig 1999), Danish (Brøndum-Nielsen 1974), Totonac (MacKay 1999), Burushaski (Grune 1998). These have the desired regularities but can be harder for the clustering to single out. The base-12 language Nungu (Mathews 1917) abstracts so well that one suspects that the source is idealized.

An interesting observation is that the algorithm frequently proceeds by finding "horizontal" abstractions, that is, it often groups `twenty-one, thirty-one, ..., ninety-one` rather than `twenty-one, twenty-two, ..., twenty-nine`. If this behaviour should be considered unwanted it's easy to penalize for for a particular language, but there's really nothing wrong with

Language	Accuracy	Language	Accuracy
af tunni	0.62	albanian	0.90
amharic	0.95	basque	0.82
bearlake slave	0.51	biblical hebrew	0.86
breton	0.85	bulgarian	0.70
catalan	0.91	classical arabic	0.22
classical greek	0.92	croatian	0.77
czech	0.40	dagur	0.97
danish	0.92	dutch	0.87
english	0.92	finnish	0.55
french	0.47	fulfulde	0.42
ge'ez	0.95	german	0.88
guahibo	0.60	guarani	0.60
hindi	0.49	hungarian	0.46
icelandic	0.88	irish	0.31
irula	0.35	italian	0.86
japanese	0.97	kabardian	0.26
kambera	0.66	kawaiisu	0.48
khmer	0.97	khowar	0.67
kodagu	0.39	kolyma yukaghir	0.36
korean	0.72	kulung	0.51
kwami	0.88	lalo	0.97
lamani	0.97	latvian	0.92
lithuanian	0.92	lotuxo	0.97
maale	0.97	malay	0.52
maltese	0.52	mapuche	0.97
margi	0.66	maybrat	0.45
miya	0.88	modern greek	0.88
mongolian	0.77	nenets	0.97
nungu	1.00	nootka	0.66
old church slavonic	0.43	oromo	0.97
pashto	0.39	polish	0.35
portuguese	0.92	quechua	0.97
romanian	0.92	russian	0.51
samoan	0.97	sango	0.97
sanskrit	0.77	slovak	0.51
sorani	0.77	spanish	0.92
stieng	0.97	supyire	0.67
swahili	0.97	swedish	0.92
swiss french	0.88	tamil	0.39
tibetan	0.35	totonac	0.43
tuda-daza	0.97	tukang besi	0.97
turkish	0.97	votic	0.67
welsh	0.51	yasin burushaski	0.67
yucatec	0.57	zaiwa	0.97

Table 9.2: Language and grammar deduction accuracy

it. Sometimes it's even necessary as in e.g Hindi where 19, 29, ..., 99 and formed by subtraction (French is reminiscent in having `-et-` only before `un` combined with tens ,as in `vingt-et-un` vs. `vingt-deux`). Hindi (McGregor 1995) is also the most irregular language in the sample in that, although there are some patterns, its forms for 11–99 are all idiosyncratic.

On the other hand, the majority of languages have very palatable number systems and can be learnt with ease. Some outstanding examples are Lalo (Björverud 1998), Sango (Samarin 1967), Japanese (University 1996), Quechua, Stieng (Thomas 1977), Mapuche (Smeets 1989), and Tukang Besi (Donohue 1999) in which there's basically just one or two rules.

4 Discussion

We have shown that a fairly simple hybrid unsupervised/analytical algorithm can abstract much of the syntax/morphology of number words ≤ 100 across languages. Much higher accuracy is of course achievable if one drops generality, and we have data to add supervised learning stages, but the generality is what makes the method interesting.

The method works well also on sets of strings that are not numeral systems and on incompletely specified numeral systems – for an input of 50 randomly chosen numbers in 1..100 in English the output is similar.

However, the main issue for wider use of the method is scalability. Suppose we want to cluster a lot more than 101 strings. The bottleneck part of the algorithm is to compute the distance between any two strings, which is $O(n^2m^2)$ for n strings of length m. The fact that there no such thing as the mean for a set of strings (as there is for vectors) also adds to the time consumption in practice in the k-means computation. Intelligent choosing for which k:s to try clustering is not a time-complexity issue. We presume that the best option for an asymptotic speed-up, still keeping the general architecture, is to replace the edit-distance with a heuristic linear-time distance measure. However, we also expect accuracy to decrease with larger input sets if the input data is "incomplete".

With only a few further assumptions, there is a clear sense in which the algorithm can say which numeral systems are more or less complicated, that is which that have most irregularities and varied formation rules. This would be hard to achieve without computers and unsound if abstraction is supervised.

Acknowledgements

We wish thank the consulted libraries of Stockholm, Uppsala and Gothenburg and their xerox-machines. A huge debt of gratitude is to Dr. Aarne Ranta and the VINNOVA project for financial support.

References

Balakrishnan, R. (1977). *A Grammar of the Kodagu*. Annamalai University, Annamalai.

Baroni, M. (2000). *Distributional Cues in Morpheme Discovery: A Computational Model and Empirical Evidence*. Ph. D. thesis, University of California, Los Angeles.

Björverud, S. (1998). *A Grammar of Lalo*. Ph. D. thesis, Lunds Universitet.

Brøndum-Nielsen, J. (1928-1974). *Gammeldansk Grammatik: i sproghistorisk fremstilling (8 bd.)*, Chapter VI, pp. 167–231. Schultz, Copenhagen. 540 – 551.

Colarusso, J. (1992). *A Grammar of the Kabardian Language*. University of Calgary Press, Canada.

de Marcken, C. (1996). *Unsupervised Language Acquisition*. Ph. D. thesis, Massachusetts Institute of Technology.

Dodds, R. W. (1977). *Teach Yourself Malay*. Hodder and Staughton, London.

Donohue, M. (1999). *A Grammar of Tukang Besi*, Volume 20 of *Mouton Grammar Library*. Mouton de Gruyter.

Evans, N. D. (1995). *A Grammar of Kayardild: With Historical-Comparative Notes on Tangkic*, Volume 15 of *Mouton Grammar Library*. Mouton de Gruyter.

Folan, W. J. (1986). Calendrical and numerical systems of the nootka. In M. P. Closs (Ed.), *Native American Mathematics*, pp. 93–108. University of Texas Press, Austin.

Ghosh, J. (2003). Scalable clustering. In N. Ye (Ed.), *The Handbook of Data Mining*, Human Factors and Ergonomics, Chapter 10, pp. 247–277. Lawrence Erlbaum Associates, Publishers, Mahwah, New Jersey.

Goldsmith, J. (2001). Unsupervised learning of the morphology of natural language. *Computational Linguistics 27*(2), 153–198.

Grune, D. (1998). Burushaski: An extraordinary language of the karakoram mountains.

Haywood, J. A. and H. M. Nahmad (1962). *New Arabic Grammar of the Written Language* (2 ed.). Harvard University Press.

Herbert Bruce, H. (1978). *A Grammar of the Tibetan Language: Literary and Colloquial*. Motilal Banarsidass, Delhi. Reprint from the 1912 edition.

Johnson, H. and J. Martin (2003). Unsupervised learning of morphology for english and inuktitut. In *HLT-NAACL 2003, Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, May 27 - June 1, Edmonton, Canada*, Volume Companion Volume - Short papers.

Kondrak, G. (2002). *Algorithms for Language Reconstruction*. Ph. D. thesis, Department of Computer Science, University of Toronto.

MacKay, C. J. (1999). *A Grammar of Misantla Totonac*. Studies in Indigenous Languages of the Americas. University of Utah Press, Salt Lake City.

Maria Mas, Joan Melcion, R. R. and M. H. Vergés (2000). *Digui, Digui: Curs de Català* (3 ed.). Publicacions de l'Abadia de Montserrat i Enciclopédia Catalana, S. A.

Mathews, H. F. (1917). Notes on the nungu tribe, nassawara province, northern nigeria, and the neighboring tribes which use the duodecimal system of numeration. *Harvard African*

Studies I, 83–93.
McGregor, R. S. (1995). *Outline of Hindi Grammar* (3 ed.). Oxford University Press.
Nandriş, G. (1959). *Handbook of Old Church Slavonic Part 1: Old Church Slavonic Grammar*. The Athlone Press, University of London.
Nyberg, H. S. (1972). *Hebreisk Grammatik* (2 ed.). Almqvist & Wiksell, Stockholm.
Penzl, H. (1955). *A Grammar of Pashto: A Descriptive Study of the Dialect of Kanahar*. Number 2 in Program in Oriental Languages Publications Series B. American Council of Learned Societies, Washington, D.C.
Perialwar, R. (1978). *A Grammar of the Irula Language*. Annamalai University, Annamalai.
Philomena Hedwig, D. (1999). *A Grammar of Maybrat: a language of Bird's Head, Irian Jaya, Indonesia*. Ph. D. thesis, Rijksuniversiteit te Leiden.
Queixalos, F. (1986). Autobiographie d'une neonumeration. *Amerindia 11*, 155–164.
Ranta, A. (2004). Grammatical framework: A type-theoretical grammar formalism. *Journal of Functional Programming 14*(2), 145–189.
Russell, S. and P. Norvig (2003). *Artifical Intelligence: A Modern Approach* (2nd International Edition ed.). Prentice-Hall, Pearson Education, New Jersey.
Samarin, W. J. (1967). *A Grammar of Sango*. Mouton de Gruyter, Den Haag.
Smeets, I. (1989). *A Mapuche Grammar*. Ph. D. thesis, Rijksuniversiteit te Leiden.
Thomas, D. (1977). South bahnaric and other mon-khmer numeral systems. *Linguistics 174*, 65–80. Etymology hints are in the introduction (pp. 5-20) of the same volume.
Trépos, P. (1980). *Grammaire bretonne*. Imprimerie Simon, Rennes.
University, I. C. (1996). *Japanese for College Students: Basic Vol 1*. Kodansha International, Tokyo.
van der Voort, H. (2000). *A Grammar of Kwaza: A description of an endangered and unclassified indigenous language of Southern Rondônia*. Ph. D. thesis, Rijksuniversiteit te Leiden.
Wagner, R. A. and M. J. Fischer (1974). The string-to-string correction problem. *Journal of the Association for Computing Machinery 21*(1), 168–173.
Yu, A. C. L. (2003). *The Morphology and Phonology of Infixation*. Ph. D. thesis, University of California, Berkeley.