Noname manuscript No.
(will be inserted by the editor)

Statistical Unicodification of African Languages

Kevin P. Scannell

Received: 1 January 2010 / Accepted: 16 November 2010

Abstract Many languages in Africa are written using Latin-based scripts, but often with extra diacritics (e.g. dots below in Igbo: ị, ọ, ụ) or modifications to the letters themselves (e.g. open vowels "e" and "o" in Lingala: ɛ, ɔ). While it is possible to render these characters accurately in Unicode, oftentimes keyboard input methods are not easily accessible or are cumbersome to use, and so the vast majority of electronic texts in many African languages are written in plain ASCII. We call the process of converting an ASCII text to its proper Unicode form *unicodification*. This paper describes an open-source package which performs automatic unicodification, implementing a variant of an algorithm described in previous work of De Pauw, Wagacha, and de Schryver. We have trained models for more than 100 languages using web data, and have evaluated each language using a range of feature sets.

Keywords Diacritic restoration · unicodification · under-resourced languages · African languages · machine learning

1 Introduction

The problem traditionally known as "diacritic restoration" in the European context involves inserting appropriate diacritics into an input text given as ASCII characters, in order to restore it to its "proper" form. In Africa, many languages use Latin-based scripts that have been extended with diacritics not found in European languages, or with variants of the Latin characters themselves (ŋ, ɔ, ɖ, ɓ, ...) which are available in Unicode but not in any of the 8-bit ISO 8859 character sets. We therefore propose extending the scope of the diacritic restoration problem to include the restoration of any ASCII text to its proper Unicode form, and dub this more general process "unicodification".

Kevin P. Scannell
Department of Mathematics and Computer Science
Saint Louis University
St. Louis, Missouri, USA
E-mail: kscanne@gmail.com

It is hard to overstate the importance of unicodification in the context of African languages, and under-resourced languages more generally. Much of modern natural language processing (NLP) relies on large corpora for training statistical models, corpora that are unavailable for most African languages. The web offers some hope, as more and more local language communities extend their presence on the web through blogs and other forms of online publishing. Unfortunately, for a variety of reasons (lack of proper keyboards, clumsy input methods, unfamiliarity with proper orthography, etc.), many texts found on the web are not written using proper Unicode characters. Automatic unicodification allows the construction of high-quality corpora from web data, thereby paving the way for the development of statistical NLP tools. Looking to the future, we expect the quality of web corpora themselves to improve, as integration of unicodification into authoring tools helps overcome both the lack of proper keyboards and any unfamiliarity with proper orthography that may exist in some language communities.

This project ties in closely with earlier work on the web crawler "An Crúbadán" [10] which has been used to produce corpora for almost 650 languages. These corpora have proved immensely valuable in developing basic technology for many under-resourced languages; they have been used in software for accessibility, predictive text for mobile devices, spell checkers and grammar checkers, machine translation engines, and even in audiometry for hearing-impaired children [1], [5]. Many other researchers working on African languages are turning to the web for valuable training data as well; see in particular [3] and [9] in this volume.

The Crúbadán corpora are also the primary source of training data for the models evaluated in this paper. Indeed, this is one of the ancillary research questions we hope to examine here: just how effective are (noisy) web corpora when used as training data for statistical NLP? Most researchers working on major languages are able to make use of high-quality corpora consisting of books, well-edited newspaper text, and the like. This is not realistic for most languages, and therefore the effectiveness of free web corpora as training data becomes an important question.

Yarowsky [18] provided a purely data-driven approach to the problem of diacritic restoration in 1994, focusing on French and Spanish. Quite a few other papers have looked at the problem of diacritic restoration for European languages (see [13], [11], [15], [16], [6]), often relying on pre-existing NLP resources such as electronic dictionaries and part-of-speech taggers. Mihalcea [7], [8] introduced a language-independent approach based on statistics at the character level, making it well-suited for under-resourced languages. De Pauw et al [17], [2], examined this approach for a number of African languages. These latter papers were the direct inspiration for the present work; in particular, [2] calls for a "further investigation [of the machine learning approach] on a larger array of African languages" which we have attempted to provide here.

2 Unicodification

The precise meaning of unicodification rests on the definition of the inverse process of *asciification*. This is a deterministic mapping from a subset of all Unicode characters into (strings of zero or more) ASCII characters (Unicode 0000-007F). For reasons of space we do not give the full specification of asciification here, but most of the mappings are self-evident. Since we are focused on Latin-based alphabets, the domain of asciification lies within the following ranges:

– 00A1-00FF: Latin-1 Supplement

– 0100-017F: Latin Extended-A
– 0180-0233: Latin Extended-B
– 0250-02AD: IPA Extensions (some: e.g. 0253 in Hausa, 0254 in Lingala, etc.)
– 02B9-02EE: Spacing Modifier Letters (map to empty string)
– 0300-0362: Combining Diacritical Marks (map to empty string)
– 1E00-1EF9: Latin Extended Additional

Note that some characters (combining diacritics) map to the empty string under asciification, and others map to more than one character (æ→ ae, ß→ ss, etc.).

One could conceivably also include standard Latin transliterations of other Unicode scripts in this framework: Cyrillic, Greek, Ethiopic (Ge'ez), etc. but we have not done so. There is also related work on diacritic restoration for Arabic script, again not considered here.

Unicodification is defined to be the (non-deterministic, language-dependent) inverse to asciification. Note that this definition is problematic for many African languages for which there is no agreed-upon "correct" orthography, and several unicodifications of the same text are possible. Therefore the evaluations performed below on "languages" should be taken with a grain of salt. Ideally, we would have trained and evaluated models according to "writing systems" [14]; e.g. for Hausa we would need to distinguish at least the following training sets: "no length or tone marks", "with tone but no length", "with tone and long vowels doubled", "with tone and long vowels with macrons", "with tone, long vowels unmarked, short vowels marked with cedilla", variants of these with the "hooked y" used in Niger, etc. We leave such extensions for future work.

3 The algorithms

Our program crams together code for training, evaluating, and unicodification into about 500 lines of Perl.[1] Some of the algorithms described below assume the existence of a lexicon for the language. The lexicon is "layered"; this means that at training time, it is possible to specify, in addition to the raw training text, a list of words known to be correct (the first layer) and also a second layer of words that are accepted as correct but perhaps reflect non-standard spellings. The third layer of the lexicon consists of words seen in the raw training text that do not appear in the first two layers.

We performed evaluations of the following algorithms (see Tables 1 and 2):

– **BL**: The baseline algorithm simply leaves all characters as ASCII. This is usually the same as a character-level unigram model since in most cases unmarked characters are more common than their marked or extended counterparts.
– **LL**: The lexicon-lookup algorithm assumes the existence of a 3-layer lexicon for the language as described above. For each ASCII word in the input text, this algorithm first finds all words in the first layer whose asciification equals the input word. If there is just one such word, this is taken as the output. If there is more than one, the most common one in the training data is taken. If there are none in the first layer, this is repeated at the second, and then third layers. If no word is found in any of the three layers, the word is left as ASCII. Note that if no clean word

[1] Source code and training data are available from `http://sourceforge.net/projects/lingala/` (under the GNU GPLv3, as the package `charlifter`), or directly from the author.

list exists (as is the case for many African languages), then the first two layers are empty and only the third layer (the words seen in the training texts) is used.
- **LL2**: This is the same as **LL**, but in ambiguous cases where more than one possible unicodification for a given word exists in the lexicon, a word-level bigram model is used to determine the output. Trigrams would have been feasible for many languages, but only a handful of these are African languages. We therefore restricted ourselves to bigrams for the sake of uniformity in the evaluation.
- **FS1**: This is the first of the character-level statistical models. It is possible to specify a feature set (**FS**) at training time, where features are character n-grams in a neighborhood of the character to be unicodified. We use the notation (p, n) for an n-gram that begins at position offset by p from the target character; so $(-3, 3)$ is the trigram preceding the character, $(+1, 3)$ is the one following it, etc. The **FS1** model uses features $(-3, 1)$, $(-2, 1)$, $(-1, 1)$, $(+1, 1)$, $(+2, 1)$, $(+3, 1)$, i.e. the three single characters on either side of the target character. This was considered in [7] for Romanian.
- **FS2**: Features $(-5, 1)$, $(-4, 1)$, $(-3, 1)$, $(-2, 1)$, $(-1, 1)$, $(+1, 1)$, $(+2, 1)$, $(+3, 1)$, $(+4, 1)$, $(+5, 1)$, i.e. five single characters on each side. This was also used in [7] and was the main approach in [2].
- **FS3**: Features $(-4, 3)$, $(-3, 3)$, $(-2, 3)$, $(-1, 3)$, $(0, 3)$, $(+1, 3)$, $(+2, 3)$. These were used in the paper [17], but instead of using them to classify the target character (as we do), they classify the three *trigrams* containing the target character and then use a voting system (two out of three) to select the best unicodification.
- **FS4**: Features $(-3, 3)$, $(-1, 3)$, $(+1, 3)$, i.e. the trigrams immediately preceding the target, centered on it, and following it. While we only report results for **FS1-FS4** in Tables 1 and 2, extensive experimentation with other feature sets has shown **FS4** to be consistently among the best-performing algorithms across languages.
- **CMB**: This algorithm uses **LL2** for words that appear in the lexicon, and for words not in the lexicon it uses the best-performing statistical algorithm for the given language among **FS1**-**FS4**.

We implemented a Naive Bayes classifier for both word-level and character-level modeling. Unicodification proceeds from left to right, treating each ambiguous character in the input as an independent classification problem (in particular, ignoring the results of any previous unicodifications). All models were trained on lowercased text, and smoothed using additive smoothing. For simplicity, the same smoothing parameter was used across all languages, independent of the size of the various training corpora, and this likely degraded performance to a certain extent. With additional time and computing power it would be an easy matter to tune the smoothing for individual languages and we plan to do this in the near future.

4 Evaluation

4.1 Experimental Setup

As mentioned in the introduction, the training corpora were assembled from the web using the Crúbadán web crawler [10]. Through manual inspection of character frequency profiles, and relying on input from native speakers, we selected only documents that use the correct Unicode characters for each language. These were then segmented into sentences, and any sentences that appeared to contain pollution (English text or

boilerplate text) were discarded. The training corpora were randomly sampled from the remaining sentences.

When an open source spell checker existed for a given language, we used it to generate a word list for the first layer of the lexicon. For a small number of morphologically-complex languages (Finnish, Estonian, etc.), the resulting word lists would have been much too large, so we kept only the generated words that also appeared somewhere in the full web corpus for the language.

Finally, we evaluated each of the eight algorithms described in section 3 using ten-fold cross validation. We report word-level accuracy for each algorithm (following [2], where it is argued that this is a more meaningful measure than character-level accuracy).

4.2 The Tables

A useful measure of the difficulty of the diacritic restoration problem for a given language (the "lexical diffusion" of the language) was introduced in [2]. Essentially this is the average number of possible unicodifications for a given ASCII form; more precisely, it is obtained as the number of words in the lexicon divided by the number of distinct word forms after asciifying the lexicon. We found, however, that estimates of the lexical diffusion depended greatly on the corpora we used, with especially inflated values coming from noisy web corpora. Lexical diffusion may also overstate the difficulty of the problem when there exist high and low frequency word pairs with the same asciification (e.g. Romanian *și* ("and") and *si* (rarely, a musical note)) – these count as much as pairs that are harder to disambiguate. For similar reasons, it is also somewhat misleading to use **BL** as a measure of difficulty, because it makes languages that have common words with diacritics (again, Romanian *și*) appear more difficult than they really are.

Instead, we report in column **LD1** the percentage of words in the training corpus that are incorrectly resolved by always choosing the most frequent candidate word as the unicodification. Like lexical diffusion, this measure increases when there are many possible unicodifications, but only in proportion to the frequency of the candidates in the corpus. This also makes **LD1** more robust with respect to noise and more stable across corpora.

In Tables 1 and 2, the column labeled "639" contains the ISO 639-3 code for the language[2], "Train" indicates the number of words in the training set, "Lex" is number of words in the lexicon (all layers), and the remaining columns represent word-level accuracy scores for the eight algorithms, computed using ten-fold cross validation.

The best-performing feature set (**FS1**-**FS4**) for each language is rendered in italics; this is the algorithm used along with **LL2** in the **CMB** column. As usual, the best-performing algorithm overall is marked in boldface (with some ties broken at hundredths of a percent).

[2] For reasons of space we have not listed the language names in the tables; see `http://www.sil.org/ISO639-3/codes.asp` for the full list.

4.3 Analysis of Results

Because we used different training sets and different machine learning algorithms (naive Bayes vs. memory-based learning), our results are not directly comparable to those reported in [2]. Nevertheless, our column **FS2** uses the same features as were used in that paper, and we observe that our results are in all cases lower than the ones in [2], sometimes much lower. This seems to be due almost entirely to noise in the web corpora we used for training. As a partial verification of this, we retrained the French model using a high-quality corpus (3.3M words from the Hansards) and obtained results comparable to [2] (86.6% vs. 88.3% in their paper) even with a fraction of the training data and a weaker learner. This gives a partial answer to the question raised in the introduction: the use of web texts for training in statistical NLP can have a substantial negative impact on system performance, and therefore the "web as corpus" community would benefit from further research on corpus-cleaning algorithms (cf. the CLEANEVAL competitions [4]).

The second takeaway message from the evaluation tables is that the trigram models perform consistently better than the models found in [2], [7], and [8]. This confirms an intuition which was based upon consideration of examples like the common Irish word *freisin* ("also"). Algorithms **FS1** and **FS2** incorrectly restore this word as *fréisin*, despite the much greater prior probability of the unaccented "e". This is due in large part to the fact that there is a greater chance of seeing an "i" if you assume the previous character is "é" than if you assume it is "e" (e.g. *féidir*, *éigin*, *léiriú*, *féin*); indeed the same is true for the "i" in position +3 (as exhibited by some of these same examples). On the other hand, the trigram model resolves this correctly because the full trigram *-isi-* almost never occurs following "é" but is quite common after "e" (*freisin*, *feisire*, *speisialta*, *seisiún*...).

A similar example in a more familiar language would be the word *traitement* in French, which **FS1** and **FS2** restore incorrectly as *traitément* in part because the probability of seeing a "t" before an "é" (*été*, *côté*, *vérité*, etc.) is much greater than the probability that a "t" precedes an "e". It is worth emphasizing that these are *conditional* probabilities: the bigram *-te-* is, in raw terms, much more common than *-té-*, but whereas "t" is the letter most likely to precede an "é" in French, it is only the fourth most common letter preceding an "e" (after "d", "l", and "r"). In contrast, all of our trigram models give the correct output *traitement*. Examples like these appear to be the rule rather than the exception across languages, and this is borne out by the data in Tables 1 and 2.

A comparison of the two trigram models (**FS3** and **FS4**) shows that **FS3** is superior when there is a small amount of training data available (in particular, dominating in Table 1), while **FS4** is generally better when more data is available.

For most languages the bigram word model utilized in **LL2** offers only a negligible increase in performance over **LL**. Not surprisingly, we see the biggest performance boost for languages with high **LD1** values (more frequent ambiguities) and large training corpora for building an accurate bigram model. **YORUBA**

Something perhaps surprising in the results is that **LL2** often outperforms the combination **CMB**. This is saying that *for words not in the lexicon*, leaving them as pure ASCII is a better option than trying to restore them statistically. This is true despite the fact that all of the statistical restoration models outperformed the baseline when evaluated on the full texts. This apparent paradox is again a consequence of using noisy web data; when the lexicon is large, most of the unseen words will

either be pollution (often English, no diacritics), or else words in the language but written incorrectly without diacritics, so leaving these as ASCII indeed leads to the best performance.

5 Applications

We foresee many applications for our software. Many of the Crúbadán corpora for African languages consist primarily of ASCII text, and for those languages which are properly written with tone marks or extended Latin characters, our application offers a way to generate large corpora in the correct orthography, automatically. Even in cases where the performance of the unicodification is not perfect, it at least minimizes the amount of manual correction needed to create a high-quality corpus. We have already done this for Lingala, a language having more than a million words of text on the web, but with the vast majority being pure ASCII. The corpus obtained by unicodifying this text was used to create the first Lingala spell checker as well as a predictive text application. This is a good illustration of how the construction of a high-quality corpus opens the door to a world of statistical NLP applications as discussed in the introduction.

A second important application is search. Someone who uses proper Unicode characters in a search query might not find results that are written in ASCII, and conversely ASCII queries will not retrieve results written in the proper encoding. The Irish language offers an extreme example of this: in the 1990's, an acute accent (*síneadh fada*) in Irish was often typed as a forward slash following the vowel (*si/neadh fada*, for example). Because of this, some of the largest repositories of Irish language material on the web are essentially invisible to the standard search engines.

A final obvious application of the software is the simplification of keyboard input. We would like to integrate our unicodification software into free text editors like Vim and OpenOffice.org, allowing users to enter text in plain ASCII and have the correct orthography appear on the screen "magically", even if they are not completely comfortable with the correct orthography, as is common among speakers of many African languages (Kikongo, Lingala, Kinyarwanda, etc.). We have recently made a start in this direction (together with an undergraduate student, Michael Schade) by creating a free web service and API for unicodification, as well as a Firefox add-on that implements this API.[3] See [12] for related work on French.

To date, we have trained the system for 115 languages, but as can be seen especially in Table 1, many models were trained with a minimum of data. We therefore welcome contributions of additional (or cleaner) training data for any of these languages. We are also keen to develop models for as many new languages as possible. There is sufficient training text available on the web for about 50 more Latin script languages (these are listed in the README in the `charlifter` package). For languages beyond these, we would welcome contributions of texts from local language communities who feel they might benefit from the software.

Acknowledgements We are grateful to Nuance Communications, and especially Ann Aoki Becker, for their support and for their ongoing commitment to developing input technology for under-resourced languages around the world. Thanks also to my student Michael Schade for

[3] See `http://accentuate.us/` for more information.

Table 1 Word level accuracy scores on plain text: African languages

639	Train	Lex	**LD1**	**BL**	**LL**	**LL2**	**FS1**	**FS2**	**FS3**	**FS4**	**CMB**
ada	14k	1.2k	4.44	62.8	93.8	93.8	87.8	87.0	*92.6*	92.5	**94.0**
aka	177k	16k	4.18	70.6	94.1	95.8	84.3	84.9	*90.3*	90.1	**95.9**
bam	342k	17k	2.60	69.8	95.2	95.4	83.7	83.2	*89.2*	89.2	**95.6**
bas	13k	1.7k	1.39	72.0	96.0	96.0	80.2	80.9	88.2	*88.3*	**96.1**
bci	15k	1.5k	4.90	59.7	92.3	92.4	75.5	74.2	*83.3*	82.8	**93.1**
bfa	12k	1.8k	0.31	76.5	97.4	97.4	84.1	84.3	*93.4*	92.4	**97.9**
bin	11k	1.5k	1.98	66.5	94.7	94.7	80.5	80.7	*92.6*	92.3	**95.9**
bum	39k	4.1k	3.65	69.6	92.4	92.4	79.5	79.1	*85.4*	85.2	**92.8**
byv	8k	1.0k	6.86	59.4	89.0	89.0	68.5	67.8	*79.7*	79.0	**89.4**
dua	36k	4.5k	7.82	74.5	88.4	**88.8**	76.0	75.4	*81.4*	80.2	88.5
dyo	12k	3.5k	1.40	78.0	92.9	92.9	78.0	79.2	*87.3*	85.0	**93.1**
dyu	10k	1.1k	0.52	72.7	97.2	97.2	84.4	84.6	*91.8*	91.4	**98.2**
efi	20k	2.9k	5.08	71.4	90.8	90.8	76.3	74.7	87.5	*88.2*	**91.5**
ewe	19k	3.2k	5.24	59.8	89.1	89.2	75.9	76.7	*82.7*	81.7	**90.5**
fon	36k	3.4k	29.81	32.3	66.1	66.1	55.0	54.8	*59.3*	59.2	**69.1**
fub	873k	49k	1.07	77.4	98.1	98.1	84.1	84.4	90.1	*90.7*	**98.3**
gaa	11k	2.0k	2.30	44.3	91.1	91.2	78.9	77.2	90.8	*90.9*	**94.6**
gba	9k	0.7k	1.58	89.6	**97.8**	**97.8**	92.2	92.1	*95.3*	95.0	97.7
guw	21k	2.3k	3.88	45.4	93.2	93.4	72.6	72.2	*86.9*	85.7	**94.2**
hau	472k	42k	0.83	93.5	97.5	**97.7**	95.0	94.4	*96.9*	96.6	97.6
her	9k	2.5k	0.06	95.5	98.7	98.7	95.5	95.5	*97.2*	96.9	**98.8**
ibo	31k	4.3k	7.48	54.7	88.6	89.5	75.0	75.8	*81.7*	81.3	**89.5**
igl	6k	1.2k	1.38	52.9	88.9	88.9	74.0	71.2	*81.8*	81.5	**90.8**
kam	19k	4.1k	1.46	48.5	89.0	89.0	79.7	78.4	88.2	*88.5*	**94.0**
kck	9k	1.5k	0.15	98.2	**99.5**	**99.5**	98.2	98.2	*99.3*	99.3	**99.5**
kik	85k	11k	2.17	49.4	93.8	93.8	75.9	76.4	*87.2*	86.8	**95.5**
kmb	11k	1.4k	2.27	90.5	96.9	**96.9**	90.5	90.5	*92.9*	92.4	96.8
kqn	23k	4.9k	0.40	97.2	**98.7**	**98.7**	97.2	97.2	*97.5*	97.2	98.7
lin	46k	102k	11.38	30.1	77.5	78.2	45.3	46.2	*66.0*	65.4	**78.3**
lol	1k	0.5k	0.19	74.0	**89.9**	**89.9**	74.8	74.8	*77.8*	77.8	89.4
loz	100k	9.4k	0.07	96.9	**99.7**	**99.7**	97.5	97.1	*98.6*	98.6	99.7
lua	64k	8.5k	0.82	97.5	98.6	**98.7**	*97.5*	*97.5*	97.3	97.4	98.5
lub	8k	1.9k	0.14	92.7	96.9	96.9	92.8	92.8	94.2	*94.7*	**96.9**
lun	7k	2.8k	0.09	87.6	93.7	93.7	93.9	92.0	*96.7*	96.4	**98.0**
mho	2k	1.1k	1.12	80.0	85.9	85.9	79.2	79.2	81.5	*81.8*	**86.0**
mos	51k	5.1k	3.30	54.3	93.4	93.4	78.6	76.6	89.1	*89.4*	**93.9**
nso	696k	35k	0.26	88.0	99.2	99.2	95.9	95.5	*98.8*	98.7	**99.4**
nya	22k	9.1k	0.32	94.9	97.0	97.2	94.9	94.9	*96.0*	95.7	**97.2**
nyk	7k	2.6k	0.07	96.4	99.6	99.6	96.4	96.4	96.7	*96.9*	**99.7**
plt	1293k	67k	1.22	93.9	97.6	**97.6**	*93.9*	93.9	91.7	93.6	97.5
sag	35k	2.2k	1.76	93.1	97.8	**97.9**	94.9	94.7	95.9	*96.2*	97.6
sba	9k	1.2k	4.34	76.2	91.8	92.0	77.8	77.8	82.6	*83.6*	**92.3**
seh	4k	1.3k	0.00	98.0	**99.3**	**99.3**	98.0	98.0	*98.4*	98.4	**99.3**
ses	28k	8.4k	1.54	89.6	96.2	**96.3**	93.6	90.6	93.8	*93.9*	96.3
tiv	36k	2.6k	3.57	93.1	96.0	**96.3**	94.1	94.0	*95.6*	95.5	96.1
tsn	171k	8.8k	0.03	98.0	98.2	98.2	99.2	99.2	99.7	***99.7***	98.2
tum	16k	4.1k	2.25	86.2	93.8	94.3	92.5	91.7	*93.4*	93.1	**95.7**
umb	35k	4.6k	0.42	95.0	**99.1**	**99.1**	95.2	95.2	97.1	*97.1*	99.0
urh	8k	1.4k	7.65	51.6	87.3	87.3	67.0	69.7	*83.4*	82.6	**87.8**
ven	136k	9.3k	0.52	89.8	97.8	**97.8**	94.0	93.5	*97.6*	97.6	97.7
vmw	7k	2.2k	1.47	90.5	95.5	95.5	90.4	90.4	92.6	*92.8*	**95.6**
wol	1238k	32k	3.27	82.1	95.8	**97.1**	86.1	85.6	91.9	*93.0*	97.1
yao	8k	2.8k	0.98	81.6	93.0	93.0	86.1	84.9	88.9	*89.0*	**95.2**
yor	5k	3.5k	11.73	17.9	**75.2**	**75.2**	48.4	42.7	*61.9*	61.6	75.2
zne	17k	2.4k	0.43	98.5	99.3	**99.3**	*98.4*	*98.4*	98.2	98.3	99.2

Table 2 Word level accuracy scores on plain text: Other languages

639	Train	Lex	**LD1**	**BL**	**LL**	**LL2**	**FS1**	**FS2**	**FS3**	**FS4**	**CMB**
afr	1052k	170k	0.44	98.6	99.4	**99.6**	98.7	*98.8*	96.7	98.1	99.3
als	993k	80k	4.26	63.9	94.3	94.5	79.1	79.7	*89.3*	88.8	**94.7**
azj	1358k	141k	0.95	31.7	93.4	93.6	62.9	61.3	*85.0*	83.7	**95.6**
bre	1150k	96k	0.51	89.2	97.1	97.1	94.1	93.9	94.2	*95.1*	**97.2**
cat	1236k	337k	1.17	83.4	96.2	**96.8**	88.0	88.3	89.0	*90.6*	96.6
ces	1098k	135k	3.71	51.8	95.3	**96.4**	60.5	59.5	80.4	*80.7*	96.1
cmn	78k	12k	9.83	7.9	81.7	81.8	44.6	44.9	*65.2*	64.5	**83.7**
csb	344k	44k	6.59	39.6	85.2	86.0	57.6	56.9	*77.7*	76.5	**87.7**
cym	1062k	383k	0.86	97.2	97.8	**98.2**	97.2	*97.2*	93.1	96.7	97.9
dan	1220k	423k	0.39	86.2	98.7	**98.7**	94.3	94.5	96.2	*96.9*	98.6
deu	2213k	423k	0.90	91.4	98.2	**98.4**	93.1	93.0	94.8	*95.7*	98.3
est	563k	115k	0.29	81.7	98.4	98.5	80.9	77.6	91.2	*93.5*	**98.6**
eus	1085k	145k	0.30	98.9	99.5	**99.5**	*98.8*	98.8	96.1	97.8	99.3
fao	1045k	147k	0.68	59.3	97.5	97.8	81.7	75.7	94.6	*94.8*	**98.5**
fin	758k	191k	0.22	77.4	97.0	**97.0**	77.9	81.4	*85.4*	84.9	96.5
fra	1593k	655k	1.78	84.0	98.0	**99.5**	85.4	84.7	88.5	*92.5*	99.5
fri	1181k	94k	0.68	92.9	98.5	**98.7**	95.4	95.5	95.0	*96.0*	98.5
fur	871k	420k	3.98	83.1	94.6	**96.2**	*88.2*	87.7	83.0	88.1	95.9
gle	1579k	463k	1.25	70.7	98.1	**98.7**	79.9	80.6	89.4	*89.5*	98.7
glg	1025k	571k	2.99	86.4	93.4	**93.7**	88.9	89.0	88.1	*91.3*	93.4
hat	1409k	35k	1.02	86.7	98.4	**98.7**	91.8	91.4	96.6	*96.6*	98.7
haw	35k	7.3k	3.02	64.6	96.0	**96.5**	83.7	83.6	91.6	*91.6*	96.2
hrv	895k	444k	0.26	84.1	98.6	98.6	89.5	89.4	94.9	*95.3*	**98.8**
hsb	348k	36k	1.43	64.7	96.6	96.9	76.4	77.2	88.2	*88.6*	**96.9**
hun	1579k	314k	2.64	53.0	94.9	95.3	60.4	60.8	80.7	*81.4*	**95.6**
isl	1422k	283k	1.14	49.8	95.2	95.3	80.8	79.3	*92.9*	92.9	**95.7**
ita	1275k	107k	1.47	95.1	97.7	**98.1**	93.8	93.0	93.2	*94.3*	97.9
kmr	1350k	113k	6.08	53.9	90.7	92.7	68.6	68.7	81.4	*82.1*	**93.4**
lav	1070k	372k	4.81	54.9	90.5	91.4	60.8	59.0	77.5	*79.2*	**92.1**
lit	800k	104k	3.64	65.0	94.7	**96.0**	65.8	59.8	80.6	*82.3*	95.9
mlt	402k	64k	0.91	76.2	97.6	97.6	93.1	93.1	96.2	*96.7*	**98.3**
mri	1185k	43k	3.01	76.6	96.5	**97.6**	85.5	83.0	91.5	*92.6*	97.5
nds	756k	63k	1.10	88.2	97.4	**97.6**	91.0	91.1	92.5	*93.9*	97.5
nld	1099k	278k	0.25	99.3	99.5	**99.5**	99.3	*99.3*	95.5	98.7	99.2
nno	1264k	334k	0.59	87.8	98.8	**99.0**	92.1	90.7	*96.5*	96.5	99.0
nob	929k	538k	0.45	87.1	98.9	**99.0**	92.8	90.5	96.5	*96.7*	99.0
pol	980k	121k	1.83	69.7	97.5	**98.1**	79.7	77.5	89.0	*90.4*	97.9
por	1163k	497k	2.32	83.5	97.0	**97.9**	89.1	89.7	86.2	*91.1*	97.7
quh	358k	61k	1.12	92.5	97.5	**97.7**	91.4	91.3	90.5	*92.3*	97.3
ron	1291k	132k	3.51	71.3	95.3	95.7	81.7	80.1	87.6	*87.8*	**97.0**
slk	1397k	159k	2.04	55.3	96.4	**96.9**	66.1	64.2	81.8	*82.4*	96.9
slv	1048k	110k	0.50	84.5	99.2	**99.5**	91.4	91.3	96.4	*96.5*	99.4
sme	719k	97k	2.34	62.5	91.5	92.0	67.6	67.6	*84.8*	83.8	**92.0**
smo	4k	1.1k	1.89	69.0	92.6	92.4	89.0	89.0	*93.0*	93.0	**94.2**
spa	902k	639k	1.31	89.2	97.8	**98.0**	90.7	90.2	92.3	*93.6*	97.6
src	692k	59k	1.10	92.2	96.9	**97.0**	92.6	92.6	91.4	*93.3*	96.7
swe	1119k	166k	1.05	75.8	97.1	97.9	79.4	77.0	90.3	*91.5*	**98.0**
tet	969k	50k	3.55	92.0	92.8	**93.4**	*91.0*	90.3	86.9	89.4	93.0
tuk	1390k	156k	7.72	62.2	87.4	87.7	71.9	71.3	*81.5*	81.3	**88.3**
tur	144k	113k	0.64	52.3	87.9	88.0	73.8	76.6	*86.4*	84.8	**92.8**
vie	3702k	39k	25.06	30.9	72.5	**91.8**	56.5	56.2	62.6	*65.9*	91.6
wln	1077k	169k	2.77	81.3	95.5	**96.1**	86.0	85.6	89.5	*90.9*	95.9

making this work much more accessible to language communities through his Firefox add-on, and to my many collaborators on the Crúbadán project for their help preparing the web corpora which were used to train the language models, especially Tunde Adegbola (Yoruba), Denis Jacquerye (Lingala), Chinedu Uchechukwa (Igbo), Thapelo Otlogetswe (Setswana), Abdoul Cisse and Mohomodou Houssouba (Songhay), and Outi Sané (Diola). Alexandru Szasz gave helpful feedback on Romanian, as did Jean Came Poulard on Haitian Creole. Finally, thanks to Guy De Pauw, Peter Wagacha, and Gilles-Maurice de Schryver for their encouragement of this work.

This paper is dedicated to the memory of my friend and collaborator on Frisian, Eeltje de Vries (1938–2008).

References

1. Meghan E. Caldwell, *Development of psychometrically equivalent speech audiometry materials for testing children in Mongolian*, M.S. Thesis, Brigham Young University, December 2009.
2. Guy De Pauw, Peter W. Wagacha, and Gilles-Maurice de Schryver, *Automatic diacritic restoration for resource-scarce languages*, In: V. Matousek and P. Mautner, eds., Proceedings of Text, Speech and Dialogue Conference 2007, 170–179 (2007).
3. Guy De Pauw, Peter W. Wagacha, and Gilles-Maurice de Schryver, *Collection and deployment of a parallel corpus English-Swahili*, Language Resources and Evaluation, this volume (2011).
4. Cédrick Fairon et al, eds., *Building and Exploring Web Corpora*, Proceedings of the 3rd Web as Corpus Workshop, Louvain-la-Neuve, Belgium (2007).
5. Valarie N. Haslam, *Psychometrically equivalent monosyllabic words for word recognition testing in Mongolian*, M.S. Thesis, Brigham Young University, August 2009.
6. Adrian Iftene and Diana Trandabăț, *Recovering diacritics using Wikipedia and Google*, In: Knowledge Engineering: Principles and Techniques, Proceedings of the International Conference on Knowledge Engineering KEPT2009, 37–40 (2009).
7. Rada Mihalcea, *Diacritics Restoration: Learning from Letters versus Learning from Words*, In: Proceedings of the Third International Conference on Intelligent Text Processing and Computational Linguistics (2002).
8. Rada Mihalcea and Vivi Nastase, *Letter Level Learning for Language Independent Diacritics Restoration*, In: Proceedings of CoNLL-2002, 105–111 (2002).
9. Steven Moran, *An Ontology for Accessing Transcription Systems*, Language Resources and Evaluation, this volume (2011).
10. Kevin P. Scannell, *The Crúbadán Project: Corpus building for under-resourced languages*, In: Building and Exploring Web Corpora, Proceedings of the 3rd Web as Corpus Workshop, 5–15 (2007).
11. Michel Simard, *Automatic Insertion of Accents in French Text*, In: Ide and Vuotilainen, eds., Proceedings of the Third Conference on Empirical Methods in Natural Language Processing, 27–35 (1998).
12. Michel Simard and A. Deslauriers, *Real-time automatic insertion of accents in French text*, Natural Language Engineering, 7:2, 143–165 (2001).
13. Thierry Spriet and Marc El-Bèze, *Réaccentuation Automatique de Textes*, In: FRACTAL 97, Besançon, (1997).
14. Oliver Streiter and Mathias Stuflesser, *Design Features for the Collection and Distribution of Basic NLP-Resources for the World's Writing Systems*, In: Proceedings of LREC 2006, Genova, Italy (2006).
15. Dan Tufiș and Adrian Chițu, *Automatic Diacritics Insertion in Romanian Texts*, In: Proceedings of the 5th International Workshop on Computational Lexicography COMPLEX '99, 185–194 (1999).
16. Dan Tufiș and Alexandru Ceaușu, *DIAC+: A Professional Diacritics Recovering System*, In: Proceedings of the Sixth International Language Resources and Evaluation (LREC'08), (2008).
17. Peter W. Wagacha, Guy De Pauw, and Pauline W. Githinji, *A Grapheme-Based Approach for Accent Restoration in Gĩkũyũ*, In: Proceedings of LREC'06, 1937–1940 (2006).
18. David Yarowsky, *A Comparison of Corpus-Based Techniques for Restoring Accents in Spanish and French Text*, In: Proceedings of the 2nd Annual Workshop on Very Large Text Corpora, 99–120 (1994).