Hunaccent: Small Footprint Diacritic Restoration for Social Media

Judit Ács and József Halmi
Department of Automation and Applied Informatics
Budapest University of Technology and Economics
judit@aut.bme.hu, halmi.jozsef.ferenc@gmail.com

Abstract

We present a language-independent method for automatic diacritic restoration. The method focuses on low computational resource usage, making it suitable for mobile devices. We train a decision tree classifier on character-based features without involving a dictionary. Since our features require at most a few characters of context, this approach can be applied to very short text segments such as tweets and text messages. We test the method on a Hungarian web corpus and on Hungarian Facebook comments. It achieves state-of-the-art results on web data and over 92% on Facebook comments. A C++ implementation for Hungarian diacritics is publicly available, support for other languages is under development.

Keywords: diacritic restoration, Hungarian, small devices, small footprint

1. Introduction

Diacritic restoration is the task of inserting missing diacritics in languages that have diacritically marked character in their orthography, but the diacritics are replaced with their corresponding Latinized grapheme for technical reasons, such as the lack of a specialized keyboard. Although human competence can easily restore diacritics real-time while reading, morphological, phonological and lexical information used by language technology is lost when accents are missing.

Most diacritic restoration methods are either dictionary-based (Kornai and Tóth, 1997) or grapheme-based (Mihalcea, 2002), (De Pauw et al., 2007). A decision list based approach was presented by (Yarowsky, 1999).

Recently, (Novák and Siklósi, 2015) presented an SMT-based approach for Hungarian combined with a morphological analyzer. They report up to 99.06% accuracy. (Zainkó and Németh, 2010) report 98% accuracy with a dictionary-based solution. Unfortunately, these systems are not available for download and components of the systems are non-free, therefore we could not reproduce them. To our knowledge, there are only two existing publicly available Hungarian diacritic restoration systems, one presented by (Kornai and Tóth, 1997), which is dictionary based, with a clever hashing solution to avoid excessive memory usage. Although the memory issues dealt with in the paper are no longer a concern, the agglutinative morphology of Hungarian still renders building a comprehensive word list very difficult. The other system is *charlifter* (Scannell, 2011).

2. Hungarian diacritics

Standard Hungarian uses 14 vowels, out of which 9 are diacritically marked in a symmetrical system (see Table 1). Five short vowels fit in the ASCII character set and two other short vowels do not. All long vowels fall outside ASCII. When Latinized, *á*, *é*, *í*, *ó* and *ú* are replaced by *a*, *e*, *i*, *o* and *u*, and *ő, ö* and *ű, ü* by *o* and *u* respectively. No consonants are diacritically marked. These vowels constitute 11.17% of Hungarian letters and more than 40% of Hungarian words contain at least one diacritic. In addition, Hungarian has two graphemes that are almost exclusive to Hungarian, *ő* and *ű*, and therefore the double acute accent is sometimes called *Hungarumlaut* by typographers.[1] The ISO 8859-2 and the Unicode character set support for *ő* and *ű* but *õ* and *û* are sometimes used as replacements for *ő* and *ű* or are mistakenly displayed since their codepoints in ISO 8859-1 correspond with the codepoints of *ő* and *ű* in ISO 8859-2. Other methods to avoid character-set confusion or deal with the lack of a non-ASCII keyboard are flying diacritics (*ő=o", ű=u"*) or telegram style (*ö=oe, ü=ue* etc.), but these conventions are less used nowadays and we do not address them in this paper.

Table 1: Hungarian vowels

short	a	e	i	o	ö	u	ü
long	á	é	í	ó	ő	ú	ű

Table 2: Diacritic statistics on 100M Hungarian words

non-whitespace tokens	94,365,073
types	2,230,835
accented ratio	40.77%
LexDif	1.017,9
ambiguous word type ratio	5.69%
non-ascii character ratio	11.333%

Table 2 illustrates vowel statistics computed on the first 100M (94M non-whitespace) tokens of the Hungarian Webcorpus (Halácsy et al., 2004; Zséder et al., 2012). More than 40% of tokens contain at least one accented vowel. *LexDif* (De Pauw et al., 2007) is the average number of orthographic alternatives per Latinized word. 5.69% of all word types have a non-unique inverse Latinized form. Table 3 lists the frequency and the Latinized form of each vowel.

3. Hunaccent

We present an ngram based approach without employing any kind of dictionary.

[1] *ő* is sometimes used in Faroese as well.

Table 3: Frequency of Hungarian diacritics and their Latinized form

Vowel	Latinized	Frequency
a		8.3616%
á	a	3.4328%
e		9.6705%
é	e	3.3895%
i		3.9559%
í	i	0.6142%
o		3.7476%
ó	o	0.9623%
ö	o	1.0095%
ő	o	0.8972%
u		0.9543%
ú	u	0.2615%
ü	u	0.5603%
ű	u	0.1894%

3.1. Data

We use the the Hungarian Webcorpus (Halácsy et al., 2004; Zséder et al., 2012). The corpus is POS tagged and we filter all tokens tagged as punctuation, but ignore the tags otherwise as we do not want to employ a POS tagger to the final system. Characters of the text are mapped to a small subset using the following preprocessing steps:

1. the text is lowercased,
2. punctuation is replaced with _ ,
3. digits are replaced with 0,
4. non-ASCII characters are replaced with *.

Reducing the number of different charcters to 29 avoids having an excessive amount of features. Accents are removed before feature extraction.

3.2. Features

We treat the diacritic restoration as 5 separate classification problems according to the 5 vowel groups (see Table 3). In each group, the vowels have the same Latinized form, ending up in three binary classification problems and two 4-way classification problems. We assume that an accented grapheme is always Latinized to the same ASCII character which is true for Hungarian, but might not apply to other languages.

Similarly to (Mihalcea, 2002), (Mihalcea and Nastase, 2002) and (De Pauw et al., 2007) our features are character ngrams in a sliding window approach. Word and sentence boundaries are converted to a single space and the sliding window treats the space as any other character. We experiment with three families of classifiers: decision tree, logistic regression, and SVM, all available in scikit-learn (Pedregosa et al., 2011). It turns out that decision trees considerably outperform logistic regression and SVM both in speed and in performance. The other advantage of decision trees is that the method is very good at identifying important features while keeping the decisions easy to interpret.

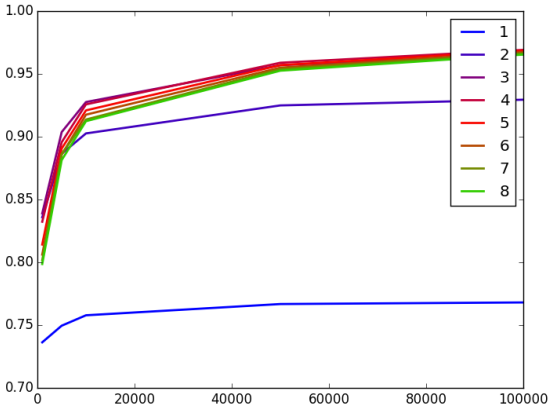

Figure 1: Average vowel accuracy with different window sizes and training data

We collect 2,000,000 occurrences of each of the 14 vowels and train on 90% and test on 10%. Hyperparameters include the sample-per-vowel count (varying from 1,000 to 2,000,000), the width of the sliding window, and the depth limit of the decision tree.

Table 4 shows the average accuracy of vowel classification taken over the 5 classification problems and weighted with the number of vowels in each class. We use symmetric sliding windows, meaning that a 4 window contains 4 characters before and 4 characters after the vowel. Accuracy improves with larger windows until a symmetric window of 4 is reached and consistently drops in each vowel class after that. These numbers are achieved without limiting the maximum depth of the decision tree. As 3 and 4 wide windows yielded the best results on 100,000 samples-per-class, we only trained with these windows on larger dataset. Memory limitation allows up to 2,000,000 samples-per-class, but in that case we had to limit the depth of the decision tree to 50.

Vowel-level accuracy is used for comparison and the best combinations are presented in Table 5.

We export the best scoring trees for each vowel group. These files are available in the *hunaccent* package and other languages will be added soon. In theory, an N deep deci-

Table 4: Vowel-level accuracy for different training sizes and window sizes

Window	Sample-per-vowel			
	5,000	10,000	50,000	100,000
1	74.95	75.78	76.68	76.8
2	88.64	90.26	92.49	92.95
3	**90.37**	**92.76**	95.69	96.55
4	89.56	92.57	**95.89**	**96.91**
5	89.05	92.1	95.7	96.85
6	88.57	91.75	95.5	96.74
7	88.62	91.37	95.38	96.63
8	88.14	91.23	95.26	96.54

Table 5: Word-level accuracy on Hungarian web corpora

Group	Depth	Sample size	Window	Acc
a	unlimited	1,000,000	4	98.92
e	50	2,000,000	4	98.17
i	50	2,000,000	4	99.63
o	unlimited	1,000,000	4	98.18
u	unlimited	500,000	4	98.61

sion tree has at most $2^{N+1} - 1$ decisions, but in practice, this number is usually much lower. The best configuration consist of 331,259 nodes for all groups, requiring a little bit over 5 MB RAM when loaded by the C++ implementation.

3.3. Accentizing social media

With a few exceptions (übra Adalı and Eryigit, 2014), diacritic restoration methods focus on well-formatted text such as newspapers or websites. As far as we know, this is the first attempt to perform it on Hungarian social media and this is why we prefer character-based methods. We used part of the Facebook comments collected by (Mihált z et al., 2015) for testing. The results are listed together with two Hungarian web corpora: the aforementioned WebCorpus and MNSZ2 (Oravecz et al., 2014)

3.4. Word-level accuracy

We compared 4 methods:

dictionary lookup retrieve the most common accentized form of every word. Leave OOV as it is. A 1,000,000 long frequency list is used.

ekito dictionary-based system by (Kornai and Tóth, 1997),

charlifter dictionary and character bigram-based system,

hunaccent our system.

Word-level accuracy was computed on a sample of 1,000,000 words on each dataset. Table 6 lists the results.

Table 6: Word-level accuracy on Hungarian web corpora

	Facebook	WebCorpus	MNSZ2
dictionary	92.67	96.98	**95.15**
ekito	92.61	94.45	93.2
charlifter	90.78	91.05	91.05
hunaccent	**92.77**	**98.36**	94.7

Table 7: Runtimes on 1M Facebook comments

Tool	Time
dictionary	26.5s
ekito	10s
charlifter	12s
hunaccent	**1.7s**

3.5. Limitations and drawbacks

The current method assumes a many-to-one mapping, where a single accented character is always mapped to single Latinized character, but more than one character may map to the same Latinized character. Tackling the issue of multicharacter mapping is out of the scope of this paper.

Another drawback of a character-based method is that non-existent word forms may be generated. Manual evaluation suggests that this is one of the largest error classes (see Section 4.).

The method does not recognize foreign words, which would be OOV in dictionary-based methods, and might accentize them (depending on the context, the English word *storage* is sometimes accentized as stóragé). A simple language model recognizing non-Hungarian words would probably help to solve this problem.

4. Manual evaluation

Manual evaluation was performed on accentized web corpora and Facebook comments using the dictionary-based and the grapheme-based methods. Considering that some words were incorrect in the input text, 4 outcomes are possible for each word: (i) correct input, correct output, (ii) incorrect input, correct output, (iii) correct input, incorrect output, (iv) incorrect input, incorrect output. Only those words were evaluated where the original and the output words differed.

The dictionary based method's errors were classified into the categories:

1. the input word is already incorrect, the output word is incorrect as well,
2. the input word is incorrect, but the diacritic restoration fixes it,
3. input word is out-of-vocabulary,
4. the input word's Latinized form is ambiguous and the wrong one is chosen from the dictionary.

Table 8 and table 9 illustrate the error classes on 1,000 manually annotated words.

Table 8: Error classes of the dictionary-based method on WebCorpus

Error class	Input	Output	Ratio
Incorrect input	írdogált	irdogalt	17.9%
Fixed input	Roviden	Röviden	8.8%
OOV	mérgesgázzal	mergesgazzal	40.5%
Ambigiuous input	feltéttel	féltettél	32.8%

Since the grapheme-based approach does not employ a dictionary, there are no OOV words, and non-existent word forms may be generated. As named entities constitute a considerable share of non-existent words, they were counted separately. Some Facebook users do not use accents, their comments were accentized and therefore differed in our output. This class is called *unaccentized input*. In some cases both the original and the output words were acceptable. Table 10 and Table 11 list the error classes on 200 samples from hunaccent's output.

Table 9: Error classes of the dictionary-based method on FB comments.

Error class	Input	Output	Ratio
Incorrect in	állapolgárok	allapolgarok	9.8%
Fixed input	boritékba	borítékba	14.75%
OOV	kormányváltók	kormanyvaltok	45.9%
Ambiguous	el	él	29.5%

Table 10: Error classes of hunaccent on MNSZ2

Error class	Input	Output	Ratio
Non-existent word	ez	éz	53%
Named entity	Theodorik	Theödorik	8%
Corrected	írdogált	irdogált	2%
Ambiguous input	igazat	igazát	36%
Incorrect input	ťhiábaŤ	ťhiábáŤ	1%

Table 11: Error classes of hunaccent on FB comments

Error class	Input	Output	Ratio
Non-existent word	ez	éz	39.5%
Named entity	Gyurcsány	Gyúrcsány	3.5%
Ambiguous word	számítana	számítaná	29%
Corrected	boritékot	borítékot	7.5%
Unaccentized input	sajat	saját	17%
Acceptable	hova	hová	3.5%

5. Conclusion and future work

We presented a small-footprint approach to diacritic restoration based on character ngrams features and using decision trees. Our experiments on Hungarian web corpora show that a symmetric 4 long sliding windows yield up to 98.36% word level accuracy and 98.88% character level accuracy when trained on a 2,000,000 sample-per-vowel dataset. We performed experiments on Hungarian Facebook comments and achieved 92.77% word-level accuracy even though the models were trained on web corpora and not social media.

Hunaccent outperforms dictionary-based approaches in all but one experiments and it is around a magnitude faster than every other tool with minimal memory footprint. The application of a moderate number of rules and a relatively short sliding window makes this approach well suited for mobile applications and social media where short texts are prevalent.

The system is available on GitHub.[2] and an Anrdoid client is under development.

De Pauw, G., Wagacha, P. W., and De Schryver, G.-M. (2007). Automatic diacritic restoration for resource-scarce languages. In *Text, Speech and Dialogue*, pages 170–179. Springer.

Halácsy, P., Kornai, A., Németh, L., Rung, A., Szakadát, I., and Trón, V. (2004). Creating open language resources for Hungarian. In *Proc. LREC2004*, pages 203–210.

Kornai, A. and Tóth, G. (1997). Gépi ékezés. *MAGYAR TUDOMÁNY*, 42(4):400–410.

Mihalcea, R. and Nastase, V. (2002). Letter level learning for language independent diacritics restoration. In *proceedings of the 6th conference on Natural language learning-Volume 20*, pages 1–7.

Mihalcea, R. F. (2002). Diacritics restoration: Learning from letters versus learning from words. In *Computational Linguistics and Intelligent Text Processing*, pages 339–348. Springer.

Miháltz, M., Váradi, T., Csertő, I., Fülöp, É., and Pólya, T. (2015). Beyond sentiment: Social psychological analysis of political facebook comments in hungary. In *Proceedings of the 6th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis (WASSA 2015)*. ACL.

Novák, A. and Siklósi, B. (2015). Automatic diacritics restoration for Hungarian. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 2286–2291.

Oravecz, Cs., Váradi, T., and Sass, B. (2014). The Hungarian Gigaword Corpus. In *Proceedings of LREC 2014*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Scannell, K. P. (2011). Statistical unicodification of African languages. *Language resources and evaluation*, 45(3):375–386.

übra Adalı, K. and Eryigit, G. (2014). Vowel and diacritic restoration for social media texts. In *Proceedings of the 5th Workshop on Language Analysis for Social Media (LASM)@ EACL*, pages 53–61.

Yarowsky, D. (1999). A comparison of corpus-based techniques for restoring accents in Spanish and French text. In *Natural language processing using very large corpora*, pages 99–120. Springer.

Zainkó, C. and Németh, G. (2010). Ékezetek gépi helyreállítása [automatic diacritic restoration]. In Géza Németh, G. O., editor, *A magyar beszéd: Beszédkutatás, beszédtechnológia, beszédinformációs rendszerek [The Hungarian Speech: Speech Research, Speech Technology]*.

Zséder, A., Recski, G., Varga, D., and Kornai, A. (2012). Rapid creation of large-scale corpora and frequency dictionaries. In *Proceedings to LREC 2012*, pages 1462–1465.

[2]https://github.com/juditacs/hunaccent