A Comparison of Letter-Level Classifiers for Portuguese Diacritic and Capitalization Restoration

Brian Rosenfeld
Princeton University
brianmr@princeton.edu

May 12, 2014

Abstract

This paper presents a comparison of letter-level, single-pass classifiers for Portuguese diacritic and capitalization restoration. It describes the results of Naive Bayes and Decision Tree classifiers trained with different windows of contextual information, ranging from one character on each side of the target letter (N=1) to three characters on each side of the target letter (N=3).

1 Introduction

Diacritic and capitalization restoration serve as important, early steps in an NLP pipeline. Addressing both in a single-pass algorithm would be an efficient approach to these two problems. Below, is an example of a proper Portuguese phrase, its diacritic and capitalization free equivalent, and their results from Google Translate. The diacritic marks affect the translation, and rightly so, for *é* is the 3rd person singular form of *to be* whereas *e* means *and*.

```
Isso é um caso de polícia. ⇒ This is a police matter.

isso e um caso de policia. ⇒ That and a case of police.
```

2 Motivation

A combined classifier for diacritic and capitalization restoration would not only offer the benefits of addressing each individual problem, but also the added advantage of serving as an integrated, one-step tool for cleaning lowercase, 7-bit ASCII text generated by speech recognizers or stripped of diacritics by 7-bit systems.

2.0.1 Diacritic Restoration

As Yarowsky[5] explained, diacritic restoration has both research and commercial applications. First, it can be used as a “front-end component to multilingual NLP systems.” Commercial uses include“use in grammar and spelling correctors, and in aids for inserting the proper diacritics automatically during on-line typing” (2). Mihalcea[4] also extols the value of diacritic restoration as part of a larger system, writing, “tools for automatic insertion of diacritics become an essential component in many important applications such as Information Retrieval, Machine Translation, Corpora Acquisition, construction of Machine Readable Dictionaries, and others.”

2.0.2 Capitalization Restoration

Similar to diacritic restoration, capitalization restoration plays an important role in preprocessing for other NLP tasks. Batista et. al[1] explain, “Besides improving human readability, punctuation marks and capitalization provide important information for parsing, machine translation (MT), information extraction, and summarization, Named Entity Recognition (NER), and further text processing tasks.”

3 Literature Survey

3.1 Diacritic Restoration

In 1999, Yarowsky [5] compared three "corpus-based techniques" for accent restoration at the word level. He found that a Decision List classifier "combines the strengths of both N-gram taggers and Bayesian classifiers, and outperforms both," explaining, "the key advantage of this decision list algorithm is that it allows the use of multiple, highly non-independent evidence types"(19-20). In 2002, Mihalcea [4], implemented a method for restoring diacritics at the character level, the first of its kind. She used a decision tree classifier and worked with Romanian, a resource-poor language. Mihalcea concluded that this model's strength is "its capability for generalization beyond words" and that it is "particularly useful for languages that lack large electronic dictionaries and morphological or syntactic tools." Like Mihalcea but unlike Yarowksy, I will be working at the letter level for diacritic restoration. Although Portuguese is not a resource-poor language, I wanted to minimize my dependency on external resources. I also liked the ability to generalize words, given that I would not be working with large datasets. Both Yarowsky and Mihalcea, mentioned the merits of decision-based classifiers, motivating me to use a Decision Tree classifier. I also implemented a Bayesian classifier, as described by Yarowsky, because this algorithm did not require addition resources such as a tagged corpus.

3.2 Capitalization Restoration

In 2004, Chelba and Acero[2] implemented a Maximum Entropy Capitalizer, treating capitalization as a tagging problem. They also noted that when adapting a classifier to a new dataset "a very small amount of domain specific data also helps significantly." In 2009, Gravano et al.[3] implemented a single-pass, n-gram model for restoring capitalization and punctuation to transcribed speech. This is similar to my single-pass model for restoring diacritics and capitalization. Lastly, in 2012, Batista et al.[1] created a two-pass approach for recovering capitalization and punctuation in Portuguese and English texts. Since all of these methods were at the word-level, I thought my classifier could offer the additional advantage of handling mixed-case words (ex. McDonald's).

4 Method

4.1 Preprocessing

To prepare the data in the .arff format used by Weka, I wrote Python scripts to preprocess the data. I used a different script for each window size (N=0, N=1, N=2, N=3). In each script, I used Python's `unidecode` module to represent Unicode data as ASCII characters. I would then convert the character to lowercase, base-7 ASCII, and use a dictionary (Table 1) for looking up the appropriate code, a value that represents the letter's diacritic and capitalization. I maintained previous characters and looked ahead for upcoming characters to provide the contextual information required for each window.

The .arff files contain enumerations of the characters considered for the classifier. This set of characters includes the lowercase letters a-z, the digits 0-9, and SPACE, PERIOD, COMMA, BOS, EOS, and OTHER. BOS and EOS refer to beginning-of-string and end-of-string respectively, and, as the name indicates, OTHER contains the set of all other characters. Initially, I used a numerical field with the ASCII representation of the character as the value, but I switched to a nominal field in order to be able to use Weka's `J48 Classifier`. The possible character codes are also enumerated for this same reason. Here is an excerpt of a .arff file with window size N=1:

```
@data
BOS, a, SPACE, 6
a, SPACE, a, 0
SPACE, a, s, 6
a, s, t, 0
s, t, r, 0
t, r, o, 0
```

Diacritic	Lower	Upper
NONE	0	6
`	1	7
´	2	8
ˆ	3	9
˜	4	10
ç	5	11

Table 1: Diacritic and capitalization pairs and their codes.

4.2 Training the Classifier

4.2.1 Tools

I trained the classifiers with the `Weka 3: Data Mining Software in Java GUI`. I trained a `NaiveBayes` and `J48` classifier each with cross-validation and 10 folds for each of the window sizes N=0, N=1, N=2, and N=3.

4.2.2 Why these classifiers?

As described earlier, Yarowsky[5] used a Bayesian classifier while both he and Mihalcea[4] used decision-based classifiers. I thought the `NaiveBayes` classifier would perform well by recognizing general patterns (ex. *o* is typically unaccented and lowercase when found two characters after *ç*). I thought `J48`, which is the Java version of the `C4.5` algorithm would perform well by recognizing common morphemes and words (ex. *ção* is a common noun ending).

4.2.3 Expectations

I expected all classifiers to outperform the baseline. I also thought that the `J48` classifiers would outperform the respective `NaiveBayes` classifiers and that performance would be increasing in window size.

4.2.4 Logistic Classifiers

I initially tried to use `Weka's Logistic` classifier because past work has suggested this model's merits for restoring capitalization; however, my first attempts with this classifier replicated the baseline, so I decided not to pursue it further.

4.3 Assumptions

4.3.1 PERIOD, COMMA, and OTHER

As mentioned previously, the only distinctions I made for characters that were neither a letter nor a digit were PERIOD, COMMA, and OTHER. I thought PERIOD deserved to be a special case because periods are usually preceded by lowercase letters and are usually followed by a space and then an uppercase letter. I thought COMMA was a special case because of its frequent use and location near lowercase letters. I placed everything else within an umbrella tag OTHER. I assumed that the characters within OTHER all had similar effects on diacritics and capitalization; however, it is most likely the case that there are finer distinctions within this category. Again, for simplicity and so that I could use a `nominal` datatype, I decided to make this assumption.

4.4 Brazilian and European Portuguese

I assumed that their was one standard version of Portuguese when there are in fact two main branches: Brazilian Portuguese and European Portuguese. In some cases, these two versions obey different forms of spelling or diacritics (ex. *cómoda* and *cômoda* respectively). This means that there could actually be two correct target codes, so it is likely that some predictions were erroneously classified as incorrect.

Window	NaiveBayes	Decision Tree
1	95.19%	96.24%
2	93.90%	97.61%
3	93.27%	97.90%

Table 2: Percent of correct letters by window size and classifier. Baseline was 94.87%.

5 Data

I used data from the most recent Portuguese Wikipedia dump (March 31, 2014). My training data consisted of 551,496 characters or 83,610 words. I initially tried to use a larger, million-character dataset but lacked the computational power to do so with Weka.

5.1 Cleaning

I used Yoichiro Hasebe's application `WP2TXT: Wikipedia to Text Convertor` to extract plain text from the Wikipedia dump file. This method was sufficiently accurate for my purposes and much more reliable than my writing scripts to extract plain text from the dump. To further clean the data, I manually attempted to remove all phonetic notation and non-Latin characters; I believe this was to a sufficient level of accuracy.

5.2 Further Assessment

My attempts to clean the data and use of `WP2TXT` removed most of the noise, but there was likely further noise in the dataset. I do not know what type of impact this had on the results. Also, as mentioned previously, the corpus was presumably a mix of European and Brazilian Portuguese. A purely Brazilian or European Portuguese corpus would have been a preferable option.

6 Results

6.1 Performance

The baseline rate for predicting both a character's diacritic and capitalization (or lack of) was 94.87%. Each Decision Tree classifier and the Naive Bayes classifier with window size N=1 outperformed the baseline. The best performance was the Decision Tree classifier with N=3. This classifier had an accuracy of 97.90%.

6.1.1 Confusion Matrices

Each Decision Tree classifier outperformed the corresponding Naive Bayes classifier. To help understand this difference, we can look at sample confusion matrices (Tables 3 & 4). The top row of each matrix shows that the Naive Bayes classifier had a higher rate of false positives, instances in which it assigned a diacritic to or capitalized a lowercase, diacritic-free character. The rates of false posities for the Naive Bayes and Decision Tree classifiers were 4.32% and 0.27% respectively. The independence assumptions of the Naive Bayes classifier explain the classifier's prolificacy in assigning diacritics. In this classifier, each individual factor may be correlated to a diacritic; however, when taken together (i.e. not independently), as in a Decision Tree classifier, this is much less likely. We can also see that the Naive Bayes classifier also has a higher rate (44.02% to 35.29%) of false-negatives, instances in which it neglected to assign a diacritic to or capitalize an uppercase or accented letter. This too can be attributed to the independence assumptions of the Naive Bayes classifier, for the Decision Tree classifier can generate rules that allow it to find specific situations in which diacritics are more likely to be needed.

6.1.2 Window Size

Decision Tree The performance of the Decision Tree classifiers increased in window size (Tables 2), improving from 96.24% for N=1 to 97.61% for N=3, representing a 44.23% decrease in the error rate.

a	b	c	d	e	f	g	h	i	j	k	l	
484442	702	5600	996	520	791	11821	1108	353	2	0	0	a
202	65	0	0	0	0	26	0	0	0	0	0	b
4526	0	2520	38	2	0	20	0	5	0	0	0	c
404	0	92	761	0	0	7	0	0	0	0	0	d
401	0	1	1	3499	0	0	0	0	0	0	0	e
424	0	0	0	0	2260	0	0	0	0	0	0	f
6048	9	7	0	3	15	5924	0	2	0	0	0	g
2	0	0	0	0	0	4	0	0	0	0	0	h
45	1	6	0	0	0	54	0	2	0	0	0	i
1	0	0	0	0	0	1	0	0	0	0	0	j
0	0	0	0	0	0	0	0	0	0	0	0	k
0	0	0	0	0	0	0	0	0	0	0	0	l

Table 3: Confusion matrix for Naive Bayes classifier with N=3.

a	b	c	d	e	f	g	h	i	j	k	l	
504980	0	564	75	28	54	625	0	9	0	0	0	a
291	0	0	0	0	0	2	0	0	0	0	0	b
2684	0	4392	27	1	0	7	0	0	0	0	0	c
487	0	10	767	0	0	0	0	0	0	0	0	d
225	0	0	1	3676	0	0	0	0	0	0	0	e
122	0	0	0	0	2560	2	0	0	0	0	0	f
5793	0	2	0	0	8	6200	0	5	0	0	0	g
1	0	0	0	0	0	5	0	0	0	0	0	h
56	0	2	0	0	0	4	0	46	0	0	0	i
2	0	0	0	0	0	0	0	0	0	0	0	j
0	0	0	0	0	0	0	0	0	0	0	0	k
0	0	0	0	0	0	0	0	0	0	0	l	

Table 4: Confusion matrix for Decision Tree classifier with N=3.

Rule Number	Prev	Letter	Next	Prediction	Instances	Accuracy
1	c	a	o	ã	1538	99.67%
2	SPACE	e	SPACE	e	3703	83.96%
3	a	c	a	ç	1270	89.89%

Table 5: Sample rules generate by Decision Tree of order N=1.

With the additional context, the Decision Tree classifier begins to recognize words, explaining the increase in accuracy. For example, the Decision Tree classifier with N=2, correctly classified every instance of the *õ* in *ações* and the *é* in *também.*

Naive Bayes The performance of the Naive Bayes classifiers worsened in window size (Tables 2), decreasing from 95.19% for N=1 to 93.27% for N=3, representing a 40.0% increase in the error rate. Again, this can be explained by the independence assumption; introducing more context introduces more noise.

6.1.3 Capitalization

The Naive Bayes and Decision Tree models with N=3 reported 49.35% and 51.67% accuracy respectively for diacritic-free uppercase letters.

6.2 Sample Tree Paths

Table 5 includes sample rules generated by the Decision Tree classifier with N=1. These rules reflect different phenomena in the Portuguese language:

1. *ção* is a common noun ending
2. *e* means *and* while *é* is the 3rd person singular form of *to be*
3. *aça* is a common noun ending (ex. *maça*) and *aca* is found in the verbs *destacar* and *acabar*

6.3 Sample Conversion of a Text

This is an excerpt from an article from *O Globo*, a Brazilian newspaper; as such, it is our reference solution:

No início, eu sofri ameaças, ligavam para a minha casa, ameaçaram a minha esposa e tive que trocar todos os telefones. Mas essas pessoas não são torcedores. Elas se aproveitavam do Cruzeiro para ganhar dinheiro.

I wrote a script for making restorations based on a classifier's predictions and ran this text with a Decision Tree (N=3) model:

No início, eu sofri ameacas, ligavam para a minha casa, ameacaram a minha esposa e tive que trocar todos os télefones. Mas essas pessoas não são torcedores. Elas se aproveitavam do Cruzeiro para ganhar dinheiro

In this example, all of the capitalization is correct, including the proper noun Cruzeiro (a Brazilian soccer team). The classifier incorrectly assigned a *c* instead of a *ç* in *ameaças* and *ameaçaram.* Both *aça* and *aca* are common in Portuguese words. Also of note (and for a reason of which I am unaware), the classifier incorrectly assigned a *é* to the first *e* in *telefones.*

6.4 Comparison to Past Results

Because this is the first implementation of a single-pass method for diacritic and capitalization restoration, there is no direct comparison to past results; however, there are smaller parallels worth exploring.

Firstly, Mihalcea[4] observed accuracies of over 99% for specific diacritic decisions. While her metric is different than the one used here, I believe that the performance of my Decision Tree classifiers (as shown by the 99.67% accuracy for the *ã* in *ção*) is somewhat comparable. Nevertheless, this example diacritic is more an exception than a standard. Like Yarowsky[5], I noticed lower performance on diacritics that reflect semantic dependencies (*e* vs. *é*). Then again, this should have been expected, for unlike Yarowsky, whose classifiers included neighboring words, the window of my classifiers were too narrow to provide enough contextual information to make semantic decisions. Lastly, when compared to the Maximum Entropy models used by Chelba and Acero[2], who reported error rates as low as 1.6%, my classifiers' performance on capitalization is astonishingly poor. Although we used different metrics for evaluating performance, the results still seem telling. In part, however, this fits Chelba and Acero's observation that "a very small amount of domain specific data also helps significantly." I believe that Wikipedia text has a relatively high number of proper nouns, which made it as if my classifiers were, at times, working out of context.

7 Future Work

Increasing Window Size Increasing the window size for a Decision Tree classifier may lead to increased performance. It would be interesting to find the optimal window size.

Variations of Portuguese Because of the differences in European and Brazilian Portuguese diacritics, it could be worthwhile to train classifiers on datasets consisting of strictly European or Brazilian Portuguese.

Logistic or Maximum Entropy Classifier This type of classifier has performed well on capitalization tasks. I am curious as to how it performs on the joint restoration of capitalization and diacritics.

Two-pass Algorithm The relatively poor performance on restoring capitalization suggests that it may be better to focus on capitalization at the word level. A model could first restore diacritics at the letter level and then in a second pass restore capitalization at the word level.

Ignoring Punctuation This would involve removing all punctuation from the training data and replacing all spaces, tabs, and other forms of white space with a BARRIER character. This would reduce the number of token, increasing the number of instances of certain patterns in the dataset.

8 Conclusion

A Decision Tree classifier is better suited to this problem than a Bayesian classifier. The independence assumption of the Naive Bayes classifier leads to a high rate of false positives whereas the rules generated by the Decision Tree classifiers allow these classifiers to be much more discerning. Additionally, the high performance of the Decision Tree classifier on certain diacritics (ex. nasal sounds such as *ã*) is encouraging. I believe that increasing the size of both the window and dataset can improve the performance on semantic-dependent diacritics. Meanwhile, the relatively poor performance on capitalization suggests that capitalization may be better suited for a word-level classifier or could be addressed by providing more contextual information. Overall, because this approach only requires a simple corpora of a sentences, I think this is a realistic model for diacritic and capitalization restoration for resource-poor languages.

References

[1] Fernando Batista, Helena Moniz, Isabel Trancoso, and Nuno Mamede. Bilingual experiments on automatic recovery of capitalization and punctuation of automatic speech transcripts. *Audio, Speech, and Language Processing, IEEE Transactions on*, 20(2):474–485, 2012.

[2] Ciprian Chelba and Alex Acero. Adaptation of maximum entropy capitalizer: Little data can help a lot. *Computer Speech & Language*, 20(4):382–399, 2006.

[3] Agustin Gravano, Martin Jansche, and Michiel Bacchiani. Restoring punctuation and capitalization in transcribed speech. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pages 4741–4744. IEEE, 2009.

[4] Rada F Mihalcea. Diacritics restoration: Learning from letters versus learning from words. In *Computational Linguistics and Intelligent Text Processing*, pages 339–348. Springer, 2002.

[5] David Yarowsky. 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, 1999.