Industry Canada
Centre for Information Technology Innovation (CITI)

Automatic Restoration of Accents in French Text

Michel Simard
simard@citi.doc.ca

Laval
January 1996

This document reports on research carried out at the Centre for Information Technology Innovation (CITI). The views expressed are strictly those of the authors.

Également disponible en français, sous le titre : “Réaccentuation automatique du français”

Catalogue no. Co28-1/129-1996E

ISBN no. 0-662-24439-7

Automatic Restoration of Accents in French Text

Michel Simard
simard@citi.doc.ca

Industry Canada
Centre for Information Technology Innovation (CITI)
1575 Chomedey Blvd.
Laval, Quebec
Canada H7V 2X2

1 Introduction

The research presented in this report is part of the *Robustness* Project conducted by the Computer-Aided Translation (CAT) team at CITI. This project is intended to develop methods and tools for the "robust" processing of natural languages: many natural-language processing (NLP) systems either refuse to process texts that contain errors or elements that are foreign to their particular knowledge base, or behave unpredictably under such circumstances. In contrast, a "robust" system will behave in a predictable and useful manner.

Unaccented French texts (i.e., texts without diacritics) are a typical and particularly common example of problems faced by NLP systems. Such problems are most often encountered in an E-mail context. Two factors account for this: First, the computer field has long suffered from a lack of sufficiently widespread standards for encoding accented characters, which has resulted in a plethora of problems in the electronic transfer and processing of French texts. Moreover, it is not uncommon for one of the software links in an E-mail distribution chain to deliberately remove accents in order to avoid subsequent problems. Secondly, keying in accented characters is still a cumbersome activity, at times requiring manual acrobatics. This is a matter of both standards and ergonomics. As a result, a large number of French-speaking users systematically avoid using accented characters, at least when writing E-mail.

If this situation remains tolerable in practice, it is essentially because it is extremely rare that the absence of accents renders a French text incomprehensible to the human reader.[1] From a linguistic point of view, the lack of accents in French simply increases the relative degree of ambiguity inhe-

1. Cases of ambiguity caused by a lack of accents do nonetheless occur. For instance, "Ce chantier ferme a cause des emeutes" could be interpreted as "Ce chantier ferme à cause des émeutes" ("This worksite is closing because of the riots") or "Ce chantier fermé a causé des émeutes" ("This closed worksite [more naturally put, this worksite closure] has caused riots").

rent in the language. At worst, it slows down reading and proves awkward, much as a text written entirely in capital letters might do.

The fact remains, however, that while unaccented French text may be tolerated under certain circumstances, it is not acceptable in common usage, especially in the case of printed documents. Furthermore, unaccented texts pose serious problems for automatic processing. Whether for purposes of documentary research, spelling, grammar or style checkers, machine translation, natural-language interface, or any other form of language processing, accents are generally required in French texts—hence the interest in methods of automatic accent restoration.

An examination of the problem reveals that the vast majority (approximately 85%) of the words in French texts take no accents, and that the correct form of more than half of the remaining words can be deduced deterministically on the basis of the unaccented form.[2] Consequently, with the use of a good dictionary, accents can be restored to an unaccented text with a success rate of nearly 95% (i.e., an error in accentuation will occur in approximately every 20 words).

All evidence suggests that much higher results can be attained through the use of moderately sophisticated *language models*, which will be able to deal with ambiguities resulting from missing accents, on the basis of linguistic considerations. More specifically, it would seem that language models termed *probabilistic* are particularly well adapted to this sort of task, as they provide a *quantitative* criterion for resolving ambiguity. When the system encounters an unaccented form that could correspond to various valid word forms (that may or may not take accents), it chooses the most *probable* one on the basis of the immediate context and a set of events observed previously (the *training corpus*).

Note, however, that this idea is not entirely original. El-Bèze et al. [3] describe an accent-restoration technique that draws upon the same concepts, while Yarowsky has obtained comparable results [6] by combining different criteria for statistical disambiguation within a unifying framework (*decision lists*).

2 Automatic Accent Restoration

We have developed an automatic accent restoration program called *Reacc*. It is based on a stochastic language model. Reacc will accept as input a string of characters that represents unaccented French text. If the input string contains accents, the accents can, of course, be stripped away. Since each accented character can correspond to only one unaccented character, this process is entirely deterministic. Another option is to retain the accents, on the assumption that they are correct. In either case, the output expected from Reacc is a string of characters that differs from the

2. In the rest of this report, *word* designates the occurrence of a linguistic unit in a given text.

input string solely in terms of accents. The expected output is therefore the same French text, the only difference being that the words are properly accented.

Reacc performs three successive operations: *segmentation*, *hypothesis generation* and *disambiguation*.

The unit on which the program operates is the *word*. Therefore, the *segmentation* process consists in taking the input string and locating word boundaries, including punctuation marks, numbers and number/letter combinations. Segmentation relies almost exclusively on language-independant data, i.e. a set of rules encoding general knowledge about the structure of electronic texts. One exception to this is a list of French abbreviations and current acronyms, which is used to determine whether a period following a string of alphabetic characters (i.e., a word) belongs to the string itself or serves to end a sentence.[3] Also, a list of the most prevalent constructions involving the hyphen and apostrophe in French is needed to determine whether or not these symbols act as word boundaries—for instance, compare *l'école* (in which case the apostrophe replaces an elided vowel in the article *la* and serves to link it with the noun that follows) with *aujourd'hui* (a single word with a compound origin), and *passe-montagne* (a compound noun) with *pensez-vous* (an interrogative inversion of pronoun and verb).

The next step, *hypothesis generation*, consists in producing a list of all accentuation possibilities for each word identified in the segmentation process. For example, if the unit *cote* has been isolated, the system would have to generate *cote, coté, côte,* and *côté.* Note that nothing precludes the generation of nonexistent words such as *côtè* and *cötê.* In practice, though, it is important to limit the number of hypotheses as much as possible so as to reduce the possibility of an undue number of combinations. The system thus draws upon a list of all valid French forms, including inflections, indexed according to their unaccented counterparts. In theory, such a list could contain several hundreds of thousands of distinct word forms. In practice, though, the number can be reduced by half, by excluding the forms that take no accents and for which there are no valid accented variants. The number can be further reduced by excluding the lower-frequency forms; however, this will eventually result in a decline in performance.

Once all the hypotheses have been generated, the most probable ones must be selected. This step is called disambiguation. A stochastic language model, called a *hidden Markov model* (HMM) is used (by means of Foster's *Im* package [4]) to carry out the disambiguation process, According to this model, a text is seen as the result of two distinct stochastic processes. The first generates a series of symbols which, in our model, correspond to morpho-syntactic tags (e.g., *CommonNoun-masculine–singular*; *Verb–indicative–present–3rdPerson–plural*). In an *N*-class HMM, the production of a symbol depends exclusively on the preceding *N-1* symbols. The sequence of tags produ-

---

3. Our heuristic excludes the possibility of a period serving simultaneously as part of an abbreviation and a marker signalling the end of a sentence. However, this does not seem to adversely affect system performance.

ced constitutes a hidden phenomenon (from which the name of the model is derived). Then for each tag in the sequence, the second process generates another symbol—in this instance, one that corresponds to a word that exists in the language. This second sequence is the observable result.

The parameters of our model are as follows:

- $P(t_i \mid h_{i-1})$: The probability of observing tag t_i, given the preceding N-1 tags (h_{i-1} designates the series of N-1 tags ending at position i-1).
- $P(f_i \mid t_i)$: The probability of observing form f_i, given the underlying tag t_i.

The exact value of these parameters is, of course, unknown, but in practice, an estimate can be made on the basis of frequencies observed in a training corpus. The corpus consists of a series of sentences, each word of which is assigned an appropriate tag (i.e., a corpus within which the nature of the "hidden" phenomenon is "revealed"). The corpus must be large enough to ensure a reliable estimate of the value of each parameter. If no such tagged corpus is available, the training operation can be carried out on an untagged text, and the parameters subsequently refined by reestimation. Another option is to combine the two methods—i.e., to obtain an initial estimate of the parameters on the basis of a small tagged corpus and then proceed with a reestimation on the basis of a larger, untagged corpus.

Given these parameters, the overall probability of a sequence of words $s = s_1 s_2 \ldots s_n$ can be evaluated. If T is the set of all possible n–length tagged sequences, then:

$$P(s) = \sum_{t \in T} \prod_{i=1}^{n} P(t_i \mid h_{i-1}) P(s_i \mid t_i)$$

Although the direct calculation of this equation requires a number of operations exponential in n, there is an algorithm that will produce the same results in polynomial time (see [5]).

Our disambiguation strategy consists in choosing a series of hypotheses that produce the version of the text with the highest overall probability. In other words, if a given text segment and its accentuation hypotheses are represented as a directed acyclic graph (DAG), the problem can be expressed as the search, from the beginning to the end of the text segment, for the pathway with the highest probability (figure 1).

Figure 1: Representation of a text segment and possible accentuation hypotheses in the form of a directed acyclic graph

There are, of course, computational complexity problems involved in the calculation of this pathway, as the number of pathways to be explored generally increases exponentially with the length of the text segment. In practice, however, it is possible to segment the graph into independent islets—i.e., into sections for which the optimal pathway is independent from the rest of the graph. Typically, sentences are considered to be independent of one another. The text can thus be segmented into sentences and the optimal pathway for each can be calculated. [4] If the number of possibilities within a given sentence remains problematic, there are ways of resegmenting the sentence, at the expense of a slight loss in accuracy. In our way of proceeding, each sentence is segmented such that the number of pathways to be explored within a given segment does not exceed a certain threshold (referred to as parameter S). The segmentation points are chosen by means of a simple heuristic that tends to minimize segment interdependence. As much as possible, each segment must end with a series of non-ambiguous words—i.e., words for which there is only one accentuation hypothesis and one lexical analysis. The segments are processed successively from left to right, and each is prefixed with the last words of the optimal pathway of the preceding segment.[5]

Once the disambiguation process has been completed, the results must be produced. Though this operation does merit attention, it is actually very simple. One of our primary concerns is to preserve the appearance of the input text once it reaches the output stage. Therefore, we must start with each word form that appears on the optimal pathway represented on the graph, find the corresponding ocurrence in the input string, and transpose the accentuation of the new form onto the original form without modifying its appearance in any other way.

3 Assessment

In order to assess the performance of an accent-restoration method, one simply has to select a French text or set of texts that are correctly accented, automatically strip them of accents, feed them into the accent-restoration program, and then compare the results with the original text.

One of the properties of Reacc that we wanted to evaluate was its ability to operate on various types of texts. In order to do so, the ideal would have been to run the program on a “balanced” corpus, along the lines of the Brown corpus. However, since no such resource was available in French, we had to construct our own corpus from the documents at our disposal.

The training corpus was therefore composed of excerpts from accented French texts drawn from seven different sources, represented in more or less equal proportions. The texts include military documents, legal texts, United Nations publications, literary texts, press reviews, computer manuals, and excerpts from Hansard (the official record of proceedings in Canada’s House of Com-

4. This notion of independence of sentences in fact becomes an integral part of the model as soon as the training itself is carried out on the basis of independent sentences.

5. An alternative is to use the Viterbi algorithm, which, for a graph such as ours, calculates the pathway that maximizes the joint probability of the sequence of word forms and the underlying tag sequence in time polynomial with the size of the graph. Since both options produce similar results, we have chosen the most direct method.

mons). The corpus totals 57,966 words (a figure produced by the UNIX *wc* utility). Certain adjustments were made to the texts in order to correct a few errors in accentuation that were found during testing.

For the purposes of our tests, the Reacc hypothesis generator used a list of word forms taken from the *Dictionnaire micro-informatisé du français* (DMF), a morpho-syntactic dictionary that contains nearly 380,000 distinct word forms [1]. Such a large number of terms is probably unnecessary. In fact, as fully satisfactory results were obtained during preliminary trials using a dictionary that recognized some 50,000 word forms only.

Where the language model was concerned, after various trials, we opted for an approach that placed a greater priority on the quality rather than the quantity of data. We used a bi-class HMM based on a set of approximately 350 morpho-syntactic tags. The parameters of the model were first initialized by means of the DMF—in other words, the $P(f_i \mid t_i)$ were restricted according to the contents of the values sanctioned by the dictionary. We then went on to an initial training phase, using a 60,000-word manually tagged corpus taken from the Canadian Hansard [2]. Lastly, a much larger untagged corpus was used, consisting of over 3 million words, in order to reestimate the model's parameters.

Aside from the hypothesis generator and the language model used, a number of other parameters affect Reacc's performance on the level of both the quality of the results obtained and running time. Nevertheless, the most important factor is the *S* parameter, which limits the size of the segments that Reacc processes. Table 1 provides the results obtained for different *S* values (an exponential increase in this factor generally translates into a linear increase in the length of the segments processed). The tests were carried out on a Sun SPARCstation 10.

Maximum no. of hypotheses per segment (*S*)	**Running time (seconds)**	**Total number of errors (words)**	**Average distance between errors (words)**
2	68	821	70
4	85	560	103
8	132	466	124
16	169	441	130
32	277	429	134
64	429	425	136
128	731	420	137

Table 1: Results of Accent-Restoration Trials

A cursory look at the results reveals that there is much to be gained by allowing the system to work on longer segments. However, beyond a certain limit, the quality of the results tends to level off, while the running time increases radically. Depending on the type of application and the resources available, it would seem that acceptable results can be obtained when S is set at around 16 or 32.

It is interesting to look at where Reacc goes wrong. Table 2 provides a rough classification of accent-restoration errors made by Reacc on our training corpus when S was set at 16. The category in which the greatest number of accentuation errors were made includes a rather liberal grouping of errors that have a common feature: they are the result of an incorrect choice pertaining to an acute accent on an *e* in the final syllable of a word (e.g., *aime* as opposed to *aimé*). The next group of errors are those that stem from inadequacies in the hypothesis generator—i.e., cases in which the generator simply does not know the correct accented form. In most cases (nearly half), proper nouns are involved, but, especially in more technical texts, there are also many abbreviations, non-French words, and neologisms (e.g., *réaménagement*, *séropositivité*). The next category concerns a unique word pair: the preposition *à*, and *a*, the third person singular form of the verb *avoir*.

Type of error	Number	Percentage
Ambiguities: *-e* vs. *-é*	171	38,8%
Unknown word forms	111	25,2%
Ambiguity: *a* vs. *à*	69	15,7%
Other	90	20,4%
Total	441	100%

Table 2: Classification of Accent Restoration Errors
(S = 16)

4 Conclusions

This report has presented a method of automatic accent restoration for French texts, based on a hidden Markov language model. This method was actually implemented by means of the Reacc software. Our experiments have demonstrated that this program produces texts that are altogether acceptable within a totally reasonable time frame: we can expect an error to occur every 130 words on average, and the processing of 20,000 words per minute.

There is, of course, always room for improvement. In particular, the use of a more refined language model (e.g., a tri-class HMM) could only enhance the quality of the disambiguation process. Moreover, given the large proportion of accentuation errors caused by words not recognized by the dictionary, it would be interesting to examine means of dealing with such “unknown” words. In this regard, we have already carried out certain preliminary experiments which have produced especially interesting results. In particular, we have focused on ways of “guessing” the accentuation of an unknown word on the basis of a stochastic model of the accentuation of known words. There is nevertheless a great deal of work to be done in this area.

The methods we described here open the door to other, similar applications. For example, we can see how generalizations could be drawn from accent-restoration methods in order to deal with other types of information loss—especially texts in which all accented characters have been replaced by a single character (most often a question mark), or texts in which the eighth bit of each character has been lost. For instance, in such texts, *é* comes out as an *i* and *è* as an *h*. In such cases, a problem of determining word boundaries compounds that of lexical ambiguity; word boundaries thus become a source of ambiguity as well.

Another interesting possibility is that of grafting a program such as Reacc onto a word processing application so that the user can input a French text without worrying about accents, which would be inserted automatically as the text is being input. This would thus mark a shift from accent restoration to automatic accentuation. This type of feature could significantly facilitate the inputting of French texts, especially in light of the lack of uniformity and ergonomic soundness in the conventions for producing accents on computer keyboards.

A much more ambitious application that could derive from similar methods is computer-assisted writing. Instead of processing a text already input by the user, the computer would be concerned with text yet to be formulated and would try to foresee what the user will type so as to avoid the need to manually input large portions of text.

All of these applications are currently being studied at CITI.

References

[1] Bourbeau, Laurent, and François Pinard. 1987. *Dictionnaire micro-informatisé du français (DMF)*. Montreal: Progiciels Bourbeau Pinard inc.

[2] Bourbeau, Laurent. 1994. *Fabrication d'un corpus témoin bilingue étiqueté et annoté pour la mise au point de techniques de parsage automatique probabiliste*. Technical report submitted by Progiciels Bourbeau Pinard to the Centre for Information Technology Innovation (CITI), Laval.

[3] El-Bèze, Marc, Bernard Mérialdo, Bénédicte Rozeron and Anne-Marie Derouault. 1994. "Accentuation automatique de textes par des méthodes probabilistes." In *Technique et sciences informatiques*, vol. 13, no. 6, pp. 797–815.

[4] Foster, George F. 1995. Personal communication.

[5] Rabiner, L. R., and B. H. Juang. 1986. "An Introduction to Hidden Markov Models." In *IEEE ASSP Magazine*, January issue.

[6] Yarowsky, David. 1994. "Decision Lists for Lexical Ambiguity Resolution: Application to Accent Restoration in Spanish and French." In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics (ACL–94)*, pp. 88–95.