From *pecher* to *pêcher*… or *pécher*: Simplifying French Input by Accent Prediction

Pallavi Choudhury **Chris Quirk** **Hisami Suzuki**
Microsoft Research
One Microsoft Way, Redmond WA 98052 USA
{pallavic, chrisq, hisamis}@microsoft.com

Abstract

In this paper we describe our approach to building a French text input system, in which no explicit typing of accents is required. This makes typing of French available to a wider range of keyboards and to occasional writers of the language. Our method is built on the noisy-channel model, and achieves 99.8% character-level accuracy on the test data consisting of formal and casual writing styles. This system is part of a larger project of making text input easier for multiple languages, including those that have traditionally been the target of input methods (such as Chinese and Japanese) as well as phonetically based non-Roman script languages (such as Greek, Russian, and Indic languages). A demo of this system including these languages will be shown at the workshop.

1 Introduction

Research on input methods has so far been limited to those languages in which an input method editor (IME) is an absolute necessity, such as Chinese and Japanese, that are written with a very large number of characters. However, with the recent availability of text prediction in typing web search queries and text on mobile devices, it has become obvious that text input methods are a very useful tool for many languages beyond the traditional IME languages (e.g., McKenzie and Tanaka-Ishii, 2007). Phonetic text input has also become widely popular in inputting non-Roman script languages in the last few years.

In this paper we deal with a problem of text input for a language that has never been a target of traditional IME: French. French uses the Roman alphabet with a few additions: accented vowels (*éàèùâêîôûëïü*), the consonant '*ç*', and the ligatures 'œ' and 'æ' Inputting these characters requires a special arrangement such as installing an international keyboard, using ALT codes (which uses the ALT key and a three or four digit code),[1] cutting and pasting from an existing text or inserting a symbol from a table. This makes French typing especially difficult for those who do not input French on a regular basis. Automatic prediction of accents should make typing faster for native speakers as well, as accented characters account for 3.64 % of French text (computed based on our training data, to be mentioned in Section 4.1).[2] Therefore, our goal is to correctly predict accents in French text within an IME scenario: users simply type characters without accents, and the accented characters are restored automatically.

We implemented our French input system based on the noisy-channel approach, which is commonly used for transliteration. Our channel model is trained using finite state transducers; our motivation for this choice will be discussed in Section 2. For language models, we use both character- and word-based n-gram models, in order to handle both contextual disambiguation of the words in the lexicon (e.g., *a* 'has' vs. *à* 'to'; *mais* 'but' vs. *maïs* 'corn') as well as accent prediction in out-of-vocabulary (OOV) words. We use a beam search decoder to find the best candidate. The models and the decoder are described in Section 3. We present our experimental results on the task of French accent prediction in Section 4, and show that our best model achieves 99.8% character-level accuracy on two test sets of different styles.

[1] On ALT codes, see http://french.about.com/od/writing/ss/typeaccents_7.htm.
[2] Rodriguez and Diaz (2007) report that in Spanish, almost half (46%) of the spelling errors (errors as the users type) are accent-related, one third of which is never corrected by the user.

Proceedings of the Workshop on Advances in Text Input Methods (WTIM 2011), pages 43–47,
Chiang Mai, Thailand, November 13, 2011.

2 Text Input as Finite State Transducer

Our IME system is built on many existing NLP technologies. In this section, we describe a method to build an IME as a finite state transducer chain using the OpenFST toolkit (Allauzen et al., 2007).

Finite state transducers have been used in transliteration for over a decade (e.g., Knight and Graehl, 1998), since they are efficient, trainable, and capture the necessary phenomena using a relatively easy-to-understand mechanism. They are a useful tool in the construction of IMEs as well. As mentioned above, for many languages and scripts the IME problem consists primarily of transliteration. In this case, a possibly weighted transducer can transform the keyboard script to the target script. Composing this with a target language model represented as a weighted acceptor will lead to more appropriate results.

Another important advantage of finite state machines is that they can represent a number of other, non-transliterating operations: physical typing errors (hitting neighboring keys by mistake), phonetic errors, and character twiddles can be represented as single transducers, and can be cascaded together to form a single machine that corrects spelling as it transliterates. Prediction can be represented by adding another machine to the cascade: a simple machine that generates any number of characters with a given weight. A single state machine where the initial state is a final state will suffice: for every character in the output script, there is an arc with an epsilon input and that character as output.

Once we have such a cascade, we can use the expectation semiring (Eisner, 2002) to train weights given parallel data. As usage of the IME increases, actual input/output pairs gathered from users can act as training data.

In a small device or a cloud service, runtime decoding with a complex FST chain may be too computationally expensive. However, we can cache some of the likely user inputs to decrease runtime computation. Likely inputs can be identified by projecting a set of common words in the target language backwards through the FST cascade offline. Say we have an IME transducer cascade of the following form: $I = misspell \circ phonetic \circ lm$. Given a set of very likely target words, such as the top K words according to unigram counts in a representative corpus, we can pack them into a simple finite state acceptor. Composing this acceptor with the inverse of the IME machine, I^{-1}, will produce a finite state machine encoding all the ways to input these words, including spelling errors, predictions, and any other operations are included in the IME cascade. For each of these likely inputs, we can compute its possible IME outputs offline and store the n-best outputs in a dictionary to avoid runtime FST complexity. Since high-frequency words tend to make up the majority of tokens, this can significantly reduce the expected runtime and therefore latency of the service. Additionally this allows us to use a complex FST chain for only certain common words, and fall back to a simpler FST at runtime for less common words.

For the French input system described in the following sections, only the basic transliteration operations are included. We now turn to the details of our model and implementation in the next section.

3 Building a French IME

3.1 Overall model structure

The French input system is based on a generalization of the noisy-channel model. In the standard noisy-channel approach, the likelihood of a target French text with accented characters t is estimated by $P(t|s) \propto P(s|t)\ P(t)$, where s is the source, unaccented sequence. The channel model $P(s|t)$ may be represented by a finite state transducer that converts characters, character sequences, or words from their unaccented and potentially misspelled forms into correctly spelled and accented French strings. Language models may take many forms; we use n-gram models over characters and words.

In the original noisy-channel model, only two equally weighted feature functions are used: the log probabilities from a channel model and a language model. We generalize this model to incorporate multiple feature functions, each with a linear weight. Currently the model has a small set of features. Word and character n-gram language models estimate the fluency of the output. For the channel model, we compute a set of likely word-based replacements offline using finite state transducers. We also allow character by character replacements, where each unaccented character may be replaced by itself or any accented variant, and characters not seen on the training set are deterministically transduced to themselves. These character replacements are currently assigned a uniform cost of –20. In the future we plan to gather parallel input and output

data, which will be useful in training a parameterized character replacement model.

3.2 Training

OpenFST is used to build our list of word-based replacements offline. A finite state transducer is used to represent the mapping from a phonetic symbol in Roman script to its orthographic symbol in the target language. For languages with different scripts, such as Greek, this would contain mappings between scripts (e.g., $r \rightarrow \rho$); in French, the characters may acquire accents during this step (e.g., $e \rightarrow \acute{e}$) or remain unchanged (e.g., $e \rightarrow e$). This character level transducer is composed with a word-level unigram language model (also represented as an FST). We shall refer to this composition as the transliteration transducer *T*. To find likely inputs in their unaccented Roman character form, we project *T* to its input domain.

For each likely input sequence, we build an FST to represent the input word and compose it with *T*. The output of the composition lists all possible potentially accented forms in the target language. We thus generate a lexicon of likely inputs in their unaccented form and their possible accented outputs in the target language. This allows us to leverage the power of an FST approach without incurring the computational cost during runtime.

3.3 Decoder

We use a beam search decoder to find the single best result according to the channel model, character n-gram language model, and replacement count features. For each prefix of the input string, we maintain the *b* best replacement candidates as scored by the weighted combination of models. We recombine hypotheses that cover the same set of input words and are indistinguishable to the character n-gram model because they share the same last n–1 characters[3].

For presenting results to the user, the efficient algorithm of Soong and Huang (1991) quickly gathers the n-best outputs. We also use this n-best list for integrating the word n-gram model. Incorporating this model directly into search requires us to score partial candidates, a somewhat complicated process since the candidates may only cover prefixes of words. Therefore, we rerank the top outputs from the system including all other features to integrate the word n-gram model. Although this integration might encounter a certain amount of search error, we find that this is seldom a problem, and the approach is both efficient and easy to implement.

4 Experiments and Results

In this section we describe the experiments we ran to evaluate the quality of French accent restoration, which serves as the basis for the French text input method. The input to the task is French text without any accent, simulating the scenario where a user types unaccented French. The output is fully accented French text. We then evaluate this output against correctly accented reference French text.

4.1 Data

For building a lexicon and training both language models, we used a collection of French corpora consisting of 840,938,412 sentences. This collection varies in style and formality, including text from news, parliamentary proceedings and web scraped documents. The lexicon built from this training corpus consists of 2,715,698 unique words.

As mentioned above, our method of training a channel model does not require any paired training data. However, we still need input/output sentence pairs for evaluating our system. For French, the creation of such paired data is easy: we just removed the accents from the target text corpus. Our test corpus consists of two sets of sentences that are disjoint from the training data: a 3,027 sentence set of news corpus from the WMT 2009 test data (WMT2009)[4] and a 5,000-sentence set from the logs of request to a machine transliteration service (MTlog). The OOV-rate of these sets against the training data is 0.44 % at the token level.

4.2 Evaluation Metric

We measured our results using character error rate (CER), which is based on the longest common subsequence match in characters between the reference and the best system output. This is a standard metric used in evaluating IME systems (e.g., Mori et al., 1998; Gao et al., 2002a,b). Let N_{REF} be the number of characters in a reference sentence, N_{SYS} be the character length of a system output, and N_{LCS} be the length of the longest common subsequence between them. Then the character-level *recall* is defined as

[3] In practice, we recombine more aggressively following the ideas in Li and Khudanpur (2008).

[4] http://www.statmt.org/wmt09/

Exp ID	Models	WMT2009			MTLog		
		b=3	b=10	b=30	b=3	b=10	b=30
E1	Channel Model Only (C.M)	0.4974	0.4974	0.4974	0.4907	0.4907	0.4907
E2	C.M + 4-gram char LM (4-CLM)	0.4643	0.4643	0.4643	0.4052	0.4052	0.4052
E3	C.M + 4-CLM + 3-gram word LM (3-WLM)	0.2499	0.2494	0.2494	0.2743	0.2740	0.2740
E4	C.M + 6-gram char LM (6-CLM)	0.3744	0.3751	0.3751	0.3322	0.3332	0.3332
E5	C.M + 6-CLM + 3-WLM	0.2389	0.2384	0.2384	0.2410	0.2418	0.2418
E6	C.M + 10-gram char LM (10-CLM)	0.2735	0.2751	0.2751	0.2448	0.2454	0.2456
E7	C.M + 10-CLM + 3-WLM	0.2183	**0.2173**	**0.2173**	0.2175	**0.2169**	**0.2169**

Table 1: Results (in CER-R, in %) of accent prediction on WMT2009 and MTLog

N_{LCS}/N_{REF}, and the *precision* as N_{LCS}/N_{SYS}. CER based on recall (CER-R) and on precision (CER-P) are then defined as 1 – recall and 1 – precision, respectively. In transliteration scenarios where the character lengths of the system output and the reference may differ (e.g., a target character corresponds to multiple source characters), CER-R and CER-P will be different. In the case of French, however, the reference and the system output are the same length in most cases (the only exceptions are the sentences that include the ligatures 'œ' and 'æ'). Therefore, we report the results using only CER-R in the next section.

4.3 Results

Table 1 show the results of French accent restoration in CER-R in two test sets, WMT2009 and MTLog, for various beam sizes (b=3, 10 and 30). Each row of the table refers to the different models we built and tested, with different combinations of the channel model, character-based and word-based language models. The first row (E1) is the baseline model which only uses the channel model. The rows E2, E4 and E6 are the systems that use the channel model and a character language model of various orders. The rows E3, E5 and E7 are the models that additionally use the word trigram model for rescoring the 50-best results of E2, E4 and E6, respectively.

From the table, we can observe that the use of a higher-order character n-gram model contributes to better accuracy consistently. Additionally, the word trigram model provides improvement over the character language model of any order. For instance, the underlined word in the following sentence is wrongly predicted as *des* by E6, but is correctly predicted by E7:

> Il est beaucoup plus important que les congressmans se mettent d'accord, dès cette semaine, qu'ils soutiennent ce plan et qu'ils le consacrerons le plus tôt possible.

Regarding beam size, it is clear from the table that a width of 10 is sufficient, as further widening does not attain any additional improvements. In addition, the beams as narrow as 3 produce quite competitive results. This is good news as speed is very important for an IME application. With a beam size of 3, it takes approximately 0.6 to 2.8 milliseconds per character depending on the complexity of the model used; this increases to 6.2 milliseconds per character with a beam size of 10.

As shown in the table, CER-R is as low as 0.22% in both test sets. This is equivalent to a character level accuracy in excess of 99.8%, which means that there is only one mistake in every 500 characters. We have also manually analyzed a sample of the remaining errors of our best model (E7) on the WMT2009 test data, and found that some (~30%) of the errors are due to ambiguous lexical entries (e.g., *Shanghai* and *Shanghaï* are both in the lexicon) and voluntary accentuation of capital letters (e.g., *Etats-Unis* and *États-Unis* are both acceptable). The remaining errors were mostly attributed to failures in contextually disambiguating the words in the lexicon (e.g., *des*/*dès*; *a*/*à*).

5 Conclusion

We have presented our system that performs French accent prediction. It achieves an accuracy of around 99.8% at the character level, which should be of great help in French input assistance, especially for non-native writers. Although accent prediction accuracy alone is not sufficient for a complete IME, it serves as a foundation for a realistic text input system.

References

Allauzen, Cyril, Michael Riley, Johan Schalkwyk, Wojciech Skut, Mehryar Mohri. 2007. OpenFst: A General and Efficient Weighted Finite-State Transducer Library. *Workshop on Implementing Automata/Conference on Implementation and Application of Automata*, pp. 11-23.

Eisner, Jason. 2002. Parameter estimation for probabilistic finite-state transducers. In *Proceedings of ACL*.

Gao, Jianfeng, Joshua Goodman, Mingjing Li and Kai-Fu Lee. 2002a. Toward a unified approach to statistical language modeling for Chinese. In *ACM Transactions on Asian Language Information Processing*, 1-1:3-33.

Gao, Jianfeng, Hisami Suzuki and Yang Wen. 2002b. Exploiting headword dependency and predictive clustering for language modeling. In *Proceedings of EMNLP*, pp.248-256.

Knight, Kevin, and Jonathan Graehl. 1998. Machine Transliteration. In *Computational Linguistics* 24(4).

Li, Zhifei and Sanjeev Khudanpur. 2008. A scalable decoder for parsing-based machine translation with equivalent language model state maintenance. In *Proceedings of ACL SSST 2008.*

McKenzie, I. Scott, and Kumiko Tanaka-Ishii. 2007. *Text Entry Systems: Mobility, Accessibility, Universality*. Elsevier.

Mori, Shinsuke, Masatoshi Tsuchiya, Osamu Yamaji and Makoto Nagao. 1998. Kana-Kanji Conversion by A Stochastic Model. SIG-NL-125-10, Information Processing Society of Japan (in Japanese).

Rodríguez, Néstor J. and Diaz, Maria I. 2007. Word processing in Spanish using an English keyboard: A study of spelling errors. In: Aykin, Nuray M. (ed.) *UI-HCII 2007 - Second International Conference on Usability and Internationalization - Part II.* pp. 219-227.

Soong, Frank, and Eng-Fong Huang. 1991. A Tree-Trellis Based Fast Search for Finding the N-Best Sentence Hypotheses in Continuous Speech Recognition. In *ICASSP*.