Error handling approach using characterization and correction steps for handwritten document analysis

Solen Quiniou, Mohamed Cheriet, Eric Anquetil

Error handling approach using characterization and correction steps for handwritten document analysis

Solen Quiniou · Mohamed Cheriet · Eric Anquetil

Abstract In this paper, we present a framework to handle recognition errors from a N-best list of output phrases given by a handwriting recognition system, with the aim to use the resulting phrases as inputs to a higher-level application. The framework can be decomposed into four main steps: phrase alignment, detection, characterization, and correction of word error hypotheses. First, the N-best phrases are aligned to the top-list phrase, and word posterior probabilities are computed and used as confidence indices to detect word error hypotheses on this top-list phrase (in comparison with a learned threshold). Then, the errors are characterized into predefined types, using the word posterior probabilities of the top-list phrase and other features to feed a trained SVM. Finally, the final output phrase is retrieved, thanks to a correction step that used the characterized error hypotheses and a designed word-to-class backoff language model. First experiments were conducted on the ImadocSen-OnDB handwritten sentence database and on the IAM-OnDB handwritten text database, using two recognizers. We present first results on an implementation of the proposed framework for handling recognition errors on transcripts of handwritten phrases provided by recognition systems.

S. Quiniou · M. Cheriet
Synchromedia Laboratory - Ecole de Technologie Superieure,
1100 rue Notre Dame Ouest, Montreal, Quebec H3C1K3, Canada
E-mail: {Solen.Quiniou, Mohamed.Cheriet}@synchromedia.ca

E. Anquetil
IRISA - INSA,
Campus de Beaulieu, 35042 Rennes Cedex, France
E-mail: Eric.Anquetil@irisa.fr

1 Introduction

With the emergence of new devices (*e.g.* smartphones, interactive whiteboards, or digital pens) and the increase in information channels, more and more multimedia data (audio, video, electronic texts, handwritten texts...) are produced days after days. In order to exploit this amount of information, high-level applications need to be designed. In recent years, applications based on natural language processing such as information retrieval, information extraction, summarization or categorization have been investigated for "clean" text documents [39]. However, multimedia data, such as blogs, SMS, e-mails, or transcripts from recognition systems (automatic speech recognition systems, handwriting recognition systems, or statistical machine translation systems) are generally "noisy". Thus, it is more difficult to apply the aforementioned techniques to these noisy data.

Recently, there has been an interest for studying problems relating to processing text data from noisy sources. Techniques from natural language processing applications thus need to be adapted to deal with such noisy data. In the field of noisy data processing, works have been focusing more particularly on dealing with text documents [39], such as blogs..., or transcripts from automatic speech recognition (ASR) systems [13, 36]. However, there have been few works that deal with transcripts from handwriting recognition (HWR) systems [26, 35]. The main issue with transcripts, whether they are from an ASR or a HWR system, is that they contain transcription errors. This is even more a problem when the transcripts are used as inputs to higher-level applications, such as information retrieval or text summarization. It is, then, of main interest to detect these recognition errors so that the higher-level system can deal with them, and/or to correct them so as to improve the performance of the higher-level system.

In this paper, we set our work in the case of handling recognition errors on handwritten transcripts, in order to make it easier for a higher-level system to process these transcripts that are given as input. Thus, we present a framework to handle recognition errors on phrase transcripts, given as output by a handwriting recognition system (here, a *phrase* may be either a sentence or a text). The proposed error handling framework consists of four steps:

1. alignment of the transcription results, given in a N-best list by the recognition system, into a word graph;
2. detection of error hypotheses on the top-list transcription, using word posterior probabilities as confidence measures (and compared to a learned threshold);
3. characterization of error hypotheses into predefined types, inherent to handwriting recognition, using a SVM and different word features (and considering different word contexts). To our knowledge, no previous work mentioned the characterization of errors in handwritten phrase recognition;
4. correction of error hypotheses, based on the error types and using a *word-to-class backoff* language model (LM), that was designed to combine efficiently a n-gram LM and a n-class LM.

In the proposed framework, we want to be as independent as possible to the recognition system used to provide the input transcripts. The only constraint on the recognizer output is that it has to be a N-best list of phrase hypotheses (usually, the list is ordered based on a recognition score); additional information on each phrase may also be given (*e.g.* its recognition score, the recognition score of each of its words, or information on the phrase segmentation into its words). Here, we use on-line handwriting recognition systems, but off-line handwriting recognition systems may also be used, or even automatic speech recognition systems (in this latter case, recognition error types may have to be changed, to reflect speech recognition errors). A preliminary version of this error handling framework was presented in [34]; here, we generalize it to consider recognizers as "black boxes", without any further assumptions on the information they provide.

The remainder of this paper is organized as follows. Sect. 2 discusses previous work on the processing of recognition errors in handwriting recognition (on-line and off-line), but also in the field of speech recognition, while Sect. 3 presents the proposed framework. Sections 4, 5, 6, and 7 describe the various parts of the framework, *i.e.* the alignment of the phrase hypotheses from a N-best list into a word graph, the detection of word error hypotheses, the characterization of these error hypotheses, and the correction of the initial top-list phrase, respectively. Sect. 8 gives an experimental evaluation of the proposed framework. Finally, some conclusions are drawn in Sect. 9.

2 Related work

In this section, we present an overview of previous works dealing with recognition errors, in the context of off-line and on-line handwriting recognition. We also review works in the field of automatic speech recognition, since most of the approaches proposed in that domain can be applied to handwriting recognition (in fact, many approaches used in handwriting recognition come from speech recognition).

2.1 Handwriting recognition

In handwriting recognition (whether off-line or on-line), there have been few work on the detection, characterization and/or correction of recognition errors, especially at the phrase level.

In fact, some handwriting recognition systems detect recognition errors by rejecting recognition results considered not sufficiently reliable. This rejection step can be performed using "anti-models" as in [27], where "anti-letter models" are used to identify incorrectly recognized words in handwritten sentences. The approach most commonly used is to compute confidence indices on words and then to compare them with a threshold to decide whether or not the words have to be rejected. In [30], several confidence measures are presented, both at the letter and word levels but no linguistic information is used in the confidence measures considered. In [3], rejection strategies based on the N-best list obtained by varying the weight of a language model are considered; the authors rely on the fact that incorrectly recognized words are more sensitive to this weight variation. However, in all these approaches, words are only detected as correct or incorrect: there is no further characterization of the errors, nor is there a correction step.

Other work has involved combining the outputs of various recognition systems: this approach can be viewed as a correction step on the output of one of these recognition systems (usually, the one that achieves the best performance). Thus, in [2], the outputs of different recognition systems are combined into a transition network and a language model is used to help retrieve the most likely output text. However, several recognition systems are needed to achieve this. In other recognition systems, the use of several language models during the recognition step can be viewed as a correction step on a baseline recognition system with only one LM. Thus, in [29, 32], a n-class LM is combined with another n-class model and with a n-gram model, respectively, whereas in [45], a stochastic context-free grammar is combined with a n-gram model: in both cases, the recognition system benefits from the use of the added language model. Finally, an original correction approach has been proposed in [12]. In this work, the correction task is considered as a translation task in which the source language is the output of the recognition system (which may contain recognition errors) and

the target language is the corrected output. Since the recognition system is considered as a black box that only gives an N-best list of candidate words for a handwritten word, the probabilities used in the correction model are estimated using the relative frequencies of each word and its translated equivalents. Nonetheless, these probabilities strongly depend on the vocabulary of the handwritten sentences and need to be trained again if the vocabulary changes. Thus, this system may not be well adapted for applications where the vocabulary is not known, *e.g.* a freeform note-taking application. In such handwritten notes, out-of-vocabulary words are more likely to occur than in notes from off-line applications where the whole vocabulary may be known in advance. Recent works also analyze the impact of recognition errors (*i.e.* "noise") on recognition tasks [26], or on higher-level tasks such as text categorization [35].

2.2 Speech recognition

In the field of automatic speech recognition, confidence measures are frequently associated with the words of the recognized outputs [1, 9–11, 17, 18, 44, 37, 42]. They are also used in machine translation tasks to assess the quality of the translation, as in [40], where various measures are compared. Among all these confidence measures, word posterior probabilities have been shown to be among the best [5, 23, 41], and they can be combined with other information sources in a neural network [23], in a SVM [21], or using conditional random fields [13] to achieve even better results. These confidence measures can also be used to detect recognition errors by rejecting words, the value of which is below a considered threshold. In [44], the error detection step is composed of two levels: first, incorrect phrases are detected (using a SVM and various features at the sentence level), and then, for the phrases detected in this way, the words are classified as either correct or incorrect (also using an SVM, and this time various features at the word level). In our proposed framework, we use these word posterior probabilities to detect recognition errors, but also to characterize them into different types (using other features in an SVM as well).

The ROVER framework [14] has been proposed in order to combine the outputs of several recognizers and then to correct the output of the best of them. From the alignment of outputs considered, this framework introduces a voting scheme to make the final choice among competing words from the various recognizers. The score given to each word combines the confidence indices given by all the recognition systems and the number of systems that outputted the word considered (in [2], this framework is extended to take a language model into account during the voting step). Furthermore, as speech recognition systems are usually multi-pass systems, they use several language models with increasing complexity in various passes, since each recognition pass

decreases the size of the search space. But, although different types of language models have been tried, n-gram models remain the most widely used. In [28], an interesting LM, called the *word-to-category backoff LM*, is presented: the category-based LM is used when the current word with its associated history is not estimated in the word-based LM (it is showed that it achieves better results than using only the word-based LM). Inspired by this work, in the final step of our framework, we combine a n-gram LM and a n-class LM (into a *word-to-class backoff LM*) to correct the errors identified. Thus, rather than combining LMs, as previous work in handwriting recognition, we use an adequate LM based on the error type of the current word.

3 Architecture of the error handling framework

Our framework, illustrated in Fig. 1, is aimed at handling recognition errors on a N-best list of phrases given by a handwriting recognition system (phrases can correspond to sentences or texts, depending on the recognition system). This framework can be divided into four parts: alignment of the input phrases, detection of the word error hypotheses on the top-list phrase, characterization of the detected error hypotheses, and correction of those errors. The various parts are presented in the following sub-sections, and they will be described in greater detail in the rest of the paper.

3.1 Alignment of the input sentences

First, the phrases of the N-best list given by the recognition system need to be aligned. For this purpose, we use an incremental alignment algorithm based on a string matching algorithm. The resulting output of this alignment module is a word graph which will be used by the following modules (see part (a) of Fig. 1). The alignment algorithm is presented in Sect. 4.

3.2 Detection of the word error hypotheses

To detect error hypotheses on the top-list phrase, a confidence index is computed for each of its words. Here, word posterior probabilities are computed from the input N-best list and are used as confidence indices, as presented in Sect. 5. Each word in the top-list phrase is finally labeled as either an error hypothesis or a correct word, by comparing its confidence index with a learned threshold (see part (b) of Fig. 1).

3.3 Characterization of the word error hypotheses

Previously detected word error hypotheses are then characterized according to predefined error types: a word with an
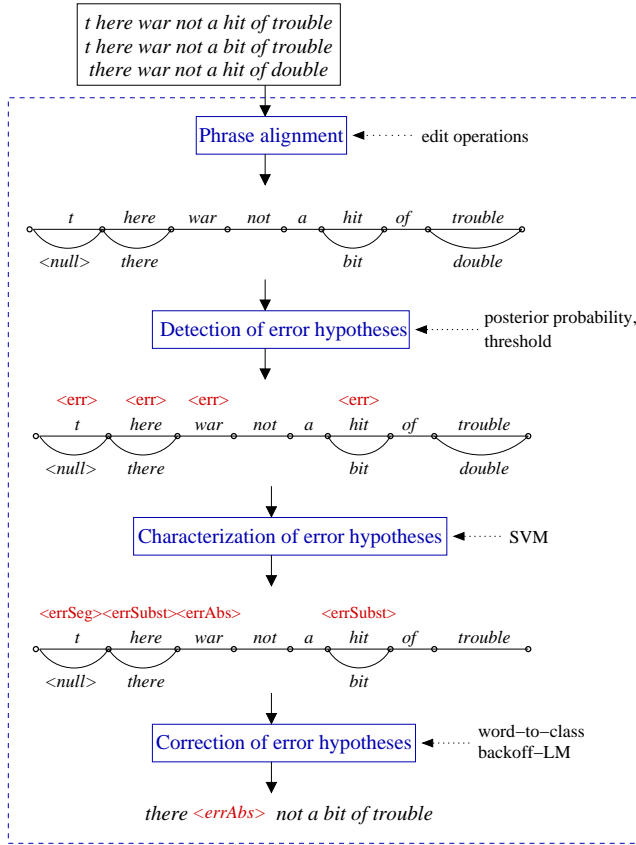
Fig. 1 Overview of the error handling framework

incorrect segmentation (*errSeg*), a word that is a substitution of the correct word that appears in the graph (*errSubst*), and a word that is a substitution of the correct word that does not appear in the graph (*errAbs*). As described in Sect. 6, the confidence index for each error hypothesis and other features are given as inputs to a SVM classifier which is learned to output the corresponding error type for each word error hypothesis. Then, each word error hypothesis of the top-list phrase is labeled according to its characterized error type. Furthermore, these error types are used to prune the word graph, so that alternative word hypotheses only remain for words identified as errors that may be corrected (see part (c) of Fig. 1).

3.4 Correction of the word error hypotheses

Finally, a language model that efficiently combines a n-gram LM and a n-class LM (called *word-to-class backoff LM*) is used on the pruned word graph to retrieve the corrected phrase. This phrase is the output of the whole framework (see part (d) of Fig. 1). The combined language model, as well as its use on the pruned word graph, is presented in Sect. 7.

4 Multiple phrase alignment using string edit operations

In order to use information from the other phrases of the N-best list, these phrases need to be aligned with the top-list phrase. A word graph is thus built. In this section, we describe how the standard edit distance is used to match pairs of phrases so as to build the word graph that represents the alignment of all the phrases in the N-best list.

The word graph is built by iteratively aligning each competing phrase of the N-best list with the top-list phrase. First, an initial word graph is built with the top-list phrase: an edge is created for each of its words. Then, for each of the other phrases in the list, the Levenshtein edit distance [24] is computed between the top-list phrase and the phrase considered: this edit distance computes the minimum number of edit operations (among *substitutions*, *insertions* and *deletions*) used to transform the phrase considered into the top-list phrase. The corresponding sequence of edit operations is used to create new edges and nodes, according to these operations (see Fig. 2):

– *substitution*: an edge labeled y_j is created, parallel to the top-list phrase edge labeled x_i (both edges have the same starting and ending nodes);
– *insertion*: an edge labeled y_j is created to be inserted between the edges labeled x_i and x_{i+1} (a new node is also created); a *null* edge is also created, parallel to the edge labeled y_j;
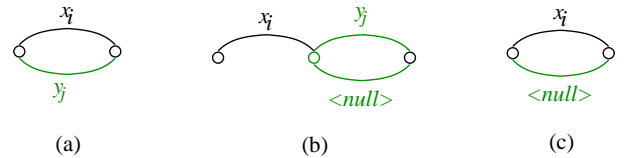– *deletion*: a *null* edge is created, parallel to the x_i edge.

Fig. 2 Examples of edge and node creation in a word graph, according to the edit operations: (a) *substitution*, (b) *insertion*, and (c) *deletion*

In the case of the substitution operation, a new edge with a label y_j is only created if it does not already exist in the word graph. If it does, information for the current word y_j, as given for the current phrase, is added to the corresponding edge (*e.g.* recognition score, where the recognition score associated with the edge becomes the maximal score among the corresponding words). In the first part of Fig. 1, there is an example of a word graph construction, from a 3-best list of phrase hypotheses.

This iterative alignment algorithm does not guarantee an optimal alignment with minimal edit costs. However, in practice, it gives an adequate solution, as a trade-off between accuracy and computational complexity.

5 Detection of error hypotheses using confidence measures on words

A common approach to detecting error hypotheses is annotating confidence at the word level. Confidence indices are thus computed for each word of the N-best phrase list, using the N-best list itself and the word graph built from it (see Sect. 4). These confidence indices are then used to detect potential recognition errors on the top-list phrase. In this section, we first describe how word posterior probabilities are computed, to be used as word confidence indices. Then, we present an approximation of these word posterior probabilities, depending on the information provided by the recognition system (as given in the N-best lists). Finally, we introduce the general error detection approach, which compares word confidence indices to a learned threshold.

5.1 Word posterior probabilities as confidence indices

The posterior probability of a word corresponds to the sum of probabilities of all the phrases of the N-best list that contains this word, at the same position. Word posterior probabilities are commonly used as confidence indices in tasks such as speech recognition or machine translation, where they were shown to be among the best confidence measures [5, 23]. Nevertheless, these probabilities are not widely used in the field of handwriting recognition, whereas they could also be useful as confidence measures on words.

The word posterior probabilities can be computed either from the N-best list of phrases or on the word graph. Here, we chose to compute them on the N-best list, by also using the word graph to obtain the alignment of the words at a given position of the phrases. The word posterior probability $P_{post}(w_i)$ of a word w_i can thus be computed according to Eq. 1:

$$P_{post}(w_i) = \frac{\sum_{w_i \in W^{(k)}} P(W^{(k)}|S)}{\sum_{W^{(k)}} P(W^{(k)}|S)} \qquad (1)$$

with $\sum_{w_i \in W^{(k)}} P(W^{(k)}|S)$ being the sum of probabilities of the phrases $W^{(k)}$ that contains w_i at the same position. $P(W^{(k)}|S)$ is the probability for the phrase $W^{(k)}$ given the signal S corresponding to the handwritten phrase. Using the Bayes formula, $P(W^{(k)}|S)$ can be rewritten as given by Eq. 2:

$$P(W^{(k)}|S) = P(S|W^{(k)}) \times P(W^{(k)}) \qquad (2)$$

which can then be rewritten as Eq. 3, using a decomposition of the phrase into its words:

$$P(W^{(k)}|S) = \prod_{j=1}^{N^{(k)}} P(s_j^{(k)}|w_j^{(k)}) \times P(w_j^{(k)}|h_j^{(k)}) \qquad (3)$$

with $P(w_j^{(k)}|h_j^{(k)})$ being the probability of word $w_j^{(k)}$ given its history $h_j^{(k)}$ in the phrase, which is given by a language model (see Sect. 7.2), and $P(s_j^{(k)}|w_j^{(k)})$ being the probability of part $s_j^{(k)}$ of the handwritten phrase, given the word $w_j^{(k)}$, which is given by the handwriting recognition system. These two probabilities, $P(s_j^{(k)}|w_j^{(k)})$ and $P(w_j^{(k)}|h_j^{(k)})$, are thus needed to compute the word posterior probabilities. Nevertheless, the handwriting recognition probabilities may not be always provided in the N-best list given by the recognition system. That is why we need to compute an approximation to these recognition probabilities so as to compute an approximation of the word posterior probabilities.

5.2 Approximation of the word posterior probabilities

In [40], the authors proposed various word-level confidence measures for machine translation. Here, we use two of these confidence measures as an approximation $\widetilde{P}(s_i|w_i)$ of the word recognition probability. Thus, we can combine this probability with the probability given by the language model to obtain an approximation $\widetilde{P}_{post}(w_i)$ of the word posterior probability.

The first approximation corresponds to the *relative frequency* of a word w_i in the N-best list of phrases and is given by Eq. 4:

$$f_{rel}(w_i) = \frac{1}{N} \sum_{k=1}^{N} \delta(w_j^{(k)}, w_i) \qquad (4)$$

with $\delta(w_j^{(k)}, w_i)$ being the Kronecker function, which equals 1 when $w_j^{(k)} = w_i$ and 0 otherwise ($w_j^{(k)}$ is the word w_j from the phrase $W^{(k)}$, this word being aligned to w_i).

The second approximation extends the first one by also taking into account the rank of each hypothesis phrase in the N-best list. This *rank-weighted frequency* of a word w_i is given by Eq. 5:

$$f_{rank}(w_i) = \frac{2}{N(N+1)} \sum_{k=1}^{N} \delta(w_j^{(k)}, w_i) \times (N+1-k) \qquad (5)$$

5.3 Error hypothesis detection by comparison with a threshold

Word error hypotheses can easily be detected by comparing their confidence index with a threshold τ_{err} (the optimal value of which is found on a validation set, as can be seen in the experiments in Sect. 8.5): if the confidence index of the considered word is below the learned threshold, the word is detected as an error hypothesis. The confidence index of

each word of the top-list phrase is used to label the top-list phrase words, as defined by Eq. 6:

$$class_{reco}(w_i) = \begin{cases} error & \text{if } Conf(w_i) < \tau_{err} \\ correct & \text{otherwise} \end{cases} \quad (6)$$

Here, we use the word posterior probabilities $P_{post}(w_i)$ (or the approximate posterior probabilities, $\widetilde{P}_{post}(w_i)$) as the confidence indices $Conf(w_i)$.

6 Characterization of error hypotheses into predefined types

In addition to detecting error hypotheses, it may be of interest to also characterize these error hypotheses into various types. This means that the error hypotheses could then be processed differently, according to their type (to try to correct them, for example). In this section, we present the error types we chose to consider. Then, we describe the various features that we are using as inputs to a SVM, to characterize each error hypothesis into its likeliest type. Finally, we show how these error types are used to prune the word graph that will then be used to perform the final correction step (see Sect. 7).
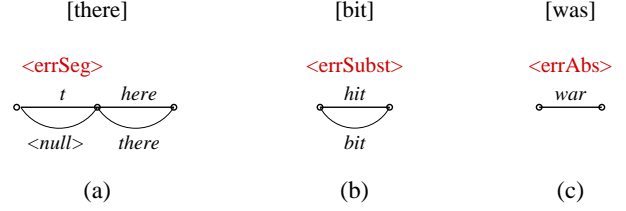
6.1 Recognition error types

In this paper, we consider three types of errors that may cause an incorrect recognition of a considered word:

– *segmentation errors* (*errSeg*): the recognition error is caused by the considered word being an incorrect segmentation part of the correct word (in part (a) of Fig. 3, *"t"* is a segmentation error of the correct word *"there"*);
– *substitution errors* (*errSubst*): the considered word is not the correct one and the correct word is one of the aligned words, corresponding to a competing edge in the word graph (in part (b) of Fig. 3, *"hit"* is a substitution error of the correct word *"bit"*);
– *absent substitution errors* (*errAbs*): the considered word is not the correct one and, as opposed to the previous error type, the correct word is not one of the alternative words. Hence, no corresponding edge appears in the word graph (in part (c) of Fig. 3, *"war"* is an absent substitution error of the correct word *"was"*).

### 6.2 Feature sets for the error characterization

To characterize the detected error hypotheses into the error types defined in the previous sub-section, we use a classifier. We have chosen to use a SVM, both because SVMs are efficient and because they are able to deal with unbalanced

Fig. 3 Examples of the error types: (a) *segmentation error*, (b) *substitution error*, and (c) *absent substitution error* (the edges corresponding to the words of the top-list phrase are shown in bold, and the correct words to recognize are given above, in square brackets)

classes (in terms of the number of training examples). The SVM is aimed at characterizing each word of the top-list phrase that has been detected as a word error hypothesis, using some features of the considered word as its inputs (each word of the top-list phrase is then further labeled with its retrieved error type).

Here, we consider various features providing different types of information such as from a graphic model, or a language model, which have been proven to be useful to detecting error hypotheses [8]. Furthermore, we consider three different feature sets, based on an incrementally enlarged context from which features are extracted for the considered word (see Fig. 4, where each context is shown with a different color): *word context* (*wSet*), *local context* (*lSet*), and *neighboring context* (*nSet*). This can be compared to work by [36], where features are also divided into different groups corresponding to different contexts of words. This is aimed at showing the improvement obtained when enlarging the word context. These three feature sets are described in the following sub-sections.
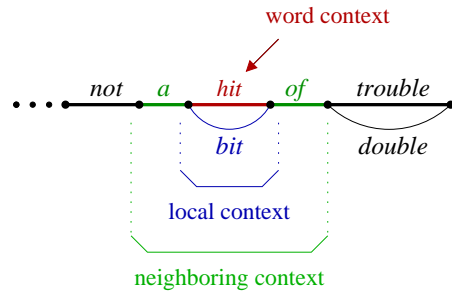
Fig. 4 Different contexts of the word *"hit"* (the edges corresponding to the words in the top-list phrase are shown in bold)

6.2.1 Word context feature set (wSet)

In this baseline set, *wSet*, we consider a context restricted to the current word itself. Hence, the features for a word w_i correspond to information on this word only. The 5 following features are considered:

- *posterior probability* (*wWordPosteriorProba*): the posterior probability $P_{post}(w_i)$ of the word, as given by Eq. 1 (or the approximated posterior probability $\widetilde{P}_{post}(w_i)$ if no recognition score is given, as explained in Sect. 5.2);
- *unigram probability* (*wWordUnigramProba*): the unigram probability of the word, given by the language model (if the word does not belong to the vocabulary associated with the language model, this probability is equal to 0);
- *length* (*wWordLength*): the length of the word (in number of characters);
- *position* (*wWordPhrasePos*): the position of the word in the top-list phrase;
- *phrase length* (*wPhraseLength*): the length of the top-list phrase (in number of words).

The first 3 features are used classically in several works on speech recognition [17, 18, 21, 44, 8] and on handwriting recognition [30] to detect recognition errors. The last 2 features were inspired by [21].

6.2.2 Local context feature set (*lSet*)

In the set *lSet*, additional information on the competing words of the current word in the word graph are considered. So, this set contains 11 features, *i.e.* the 5 previous ones as well as the following 6 new features:

- *number of competing words* (*lConcurrNbWords*): the number of competing words in the graph that have the same segmentation as the considered word (*i.e.* their edges have the same starting and ending nodes as the word considered);
- *competing null edge* (*lConcurrNullEdge*): a Boolean feature indicating whether or not the *null* edge is one of the competing words;
- *posterior probability mean* and *variance* (*lConcurrPosteriorProbasMean* and *lConcurrPosteriorProbasVariance*): the mean and variance of the posterior probabilities of all the competing words of the word considered;
- *unigram probability mean* and *variance* (*lConcurrUnigramProbasMean* and *lConcurrUnigramProbasVariance*): the mean and variance of the unigram probabilities of all the competing words of the word considered.

The mean and variance of some score features are frequently used. The use of the first feature was again inspired by [21]. In that work, they also have features similar to the second one here, but they used it for the competing words of the previous and next words to consider (*i.e.* to know whether the competing edges of the previous and next considered words contain a *null* edge). So, we add this kind of feature, but by considering the current competing edges, anticipating that it may be helpful for characterizing segmentation errors.

6.2.3 Neighboring context feature set (*nSet*)

Finally, we extend the set *lSet* to the set *nSet*, in which we also consider information on the neighboring words (the previous one and the next one in the top-list phrase). Indeed, a recognition error can often lead to a recognition error on a neighboring word. This last set contains 16 features, *i.e.* the 11 previous ones and the following 5 new features:

- *bigram probability* (*nWordBigramProba*): the bigram probability of the word, given its previous word in the top-list phrase (if the considered word does not belong to the vocabulary associated with the language model, this probability is equal to 0, and, if the previous word does not belong to the vocabulary, this probability is equal to the unigram probability of the considered word);
- *previous posterior probability* (*nPrevWordPosteriorProba*): the posterior probability of the previous word;
- *next posterior probability* (*nNextWordPosteriorProba*): the posterior probability of the next word;
- *error on the previous word* (*nPrevWordError*): a Boolean feature indicating whether or not the previous word has been detected as an error hypothesis;
- *error on the next word* (*nNextWordError*): a Boolean feature indicating whether or not the next word has been detected as an error hypothesis.

The first feature is an extension of the *unigram probability* of the baseline set, to include the context of the previous word. The second and third features were inspired by [8, 17], while the last two features are based on our previous work [34].

6.3 Word graph pruning by the error characterization types

The word graph is finally pruned according to the type of each word in the top-list phrase (a correct label or a characterized error type):

- *segmentation error*: the edge x_i of the current word is kept, as well as the edges y_i of the competing words;
- *substitution error*: the edge x_i of the current word is kept, as well as the edges y_i of the competing words;
- *absent substitution error*: only the edge x_i of the current word is kept in the graph (competing edges are pruned) and its word label is replaced by *errAbs*;
- *correct word*: only the edge x_i of the correct word is kept in the graph (competing edges are pruned).

In Fig. 5, it can be seen that the word *"double"* has been removed from the graph, as the corresponding word of the top-list phrase (word *"trouble"*) has been detected as a correct word.

The pruning step is performed to decrease the size of the graph, which in turn will reduce the complexity of the correction step that uses this graph. The pruning mainly consists of removing edges that compete with words detected
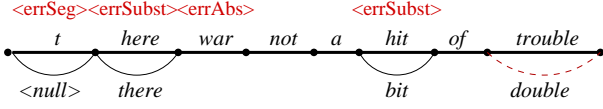
Fig. 5 Example of a pruned word graph (the pruned edges are shown with dotted lines)

as *correct* ones, thereby ensuring that their initial recognition will not change. Competing edges of error hypotheses characterized as *absent substitution errors* are also removed, ensuring that their incorrect recognition will not have as bad an impact as before on the recognition of the neighboring words. Indeed, this kind of error leads to the substitution of the current word by a wrong word, which has been shown to cause recognition errors on neighboring words, essentially due to the use of a language model during the recognition process [4]. In fact, only competing edges of *segmentation* and *substitution errors* remain in the graph (in addition to edges corresponding to words of the top-list phrase), which will be used to correct the top-list phrase.

7 Correction of error hypotheses using a word-to-class backoff language model

The characterization of error hypotheses into different types can now be used to try to correct the top-list phrase, depending on its characterized words. So, the pruned word graph can be exploited to retrieve the corresponding corrected phrase. In this section, we introduce the Maximum A Posteriori (MAP) approach that is used to find the likeliest sentence in a word graph, using a language model. Then, we briefly recall the concept of statistical language modeling. Finally, we describe the language model used in this correction step that combines a n-gram LM and a n-class LM, from its creation to its use on the pruned word graph.

7.1 Post-processing correction using a language model

To retrieve the corrected phrase that is given as the output of the whole error handling framework, we use the classic MAP decoding. This decoding is aimed at finding the likeliest phrase $\hat{W}_{correct}$ among the phrases $W^{(k)} = w_1^{(k)} \ldots w_{N_k}^{(k)}$, given the handwritten signal S (it is efficiently performed on the word graph with the Viterbi algorithm [15]):

$$\hat{W} = \arg\max_{W^{(k)}} P(S|W^{(k)}) \times P(W^{(k)})^\gamma \times N^{(k)^\delta} \quad (7)$$

with $P(S|W^{(k)})$ being the probability of the handwritten signal S for the given sentence $W^{(k)}$ (it is classically given by the recognition system), $P(W^{(k)})$ the probability of the phrase $W^{(k)}$ given by a language model and weighted by γ, and $N^{(k)}$ the number of words in the sentence $W^{(k)}$ that

is weighted by δ. Eq. 7 extends Eq. 2 by weighting each probability and by also taking into account the number of words in the phrase. Eq. 7 can be further decomposed for each word of the sentence, as given by Eq. 8:

$$\hat{W} = \arg\max_{W^{(k)}} \prod_{j=1}^{N^{(k)}} P(s_j^{(k)}|w_j^{(k)}) \times P(w_j^{(k)}|h_j^{(k)})^\gamma \times \delta \quad (8)$$

where the probability $P(s_j^{(k)}|w_j^{(k)})$ is either given by a handwriting recognition system or computed as an approximation (as explained in Sect. 5.2), and the probability $P(w_j^{(k)}|h_j^{(k)})$ is given by a language model. This language model can be a simple model (as the ones described in Sect. 7.2), or a more complex one (as the combined language models presented in Sect. 7.3).

7.2 Statistical language modeling

Before presenting the combined LM, we recall the principle of statistical language modeling and the two models most commonly used: n-gram LMs and n-class LMs.

7.2.1 General definition

Statistical language modeling is aimed at capturing the regularities of a language by the use of statistical inference on a corpus of that language. The probability of a sequence of n words $W = w_1^n = w_1 \ldots w_n$ is thus given by equation 9:

$$P(W) = \prod_{i=1}^{n} P(w_i|h_i) \quad (9)$$

where $h_i = w_1 \ldots w_{i-1}$ is called the *history* of word w_i. In practice, there are too many different histories, which leads to a tremendous number of probabilities to estimate. Furthermore, most of these probabilities occur too infrequently in the corpus to be estimated reliably. A solution would be to merge histories into equivalence classes, which results in n-gram LMs.

7.2.2 n-gram language models

In n-gram LMs, histories ending with the same $n-1$ words are considered to belong to the same equivalence class. Eq. 9 can thus be rewritten into Eq. 10:

$$P(W) = \prod_{i=1}^{n} P_w(w_i|w_{i-n+1}^{i-1}) \quad (10)$$

where n is called the *order* of the LM. The n-gram probabilities $P_w(w_i|w_{i-n+1}^{i-1})$ are estimated using relative frequencies obtained from a text corpus. Hence, the probability estimations depend on the corpus, and the probabilities of non-occurring n-grams (*i.e.* sequences of n words) will be

estimated to be zero. One way to overcome this problem is to apply a *smoothing* to the n-gram LM probabilities. The principle of the smoothing is first to reduce the probabilities of the n-grams occurring in the corpus and then to redistribute this mass of probabilities to unseen n-grams. Here, we use the Kneser-Ney smoothing, which has been shown to be very efficient [19]. Nonetheless, when words are out of the vocabulary associated with the LM, their probabilities will remain equal to zero. In that case, a solution may be to use n-class language models, where words are grouped into equivalence classes. Thus, if we can find the class of an out-of-vocabulary word, its linguistic probability will not be equal to zero.

7.2.3 n-class language models

Depending on the approach used to create the considered classes (using a statistical criterion or by considering predefined categories), a word may belong to one or more classes. For example, when considering the grammatical nature of words (also called *Part-Of-Speech* or POS tags), which is our concern here, a word may belong to several classes, the correct one depending on the context of the word. Two approaches can be used to take into account the various possible classes of words: either consider all the possible class sequences associated with a given word sequence or only consider the likeliest class sequence. We chose the latter approach, so that we could retrieve the class of an *unknown word* (an OOV word or a word characterized as an *absent substitution error*, for example). The probability $P(w_i|h_i)$ is thus based not only on the words but also on their classes, as given by Eq. 11:

$$P_c(w_i|w_{i-n+1}^{i-1}) = \max_{c_{i-n+1}^i \in C_{i-n+1}^i} P(w_i|c_i)P(c_i|c_{i-n+1}^{i-1}) \quad (11)$$

with $C_{i-n+1}^i = C_i \times \ldots \times C_{i-n+1}$, and c_j being a class in the class set C_j associated with the word w_j.

In conclusion, n-gram LMs are more accurate than n-class LMs but the latter have better generalization power. This is why we have combined efficiently these two types of language models into what is called a *word-to-class backoff LM*, as presented in the next sub-section.

7.3 Word-to-class backoff language model

In this sub-section, we first describe how a n-gram language model is efficiently combined with a n-class language model to create the *word-to-class backoff* language model and then we present how to use this new language model on the pruned word graph, to correct the top-list phrase.

7.3.1 Definition of the word-to-class backoff language model

As seen in the previous sub-sections, n-gram LMs are more accurate to provide a linguistic probability to a word, but this word and the words in its history have to belong to the vocabulary associated with the language model. In the case when the word or one word of its history is out of the vocabulary, it may be of interest to use instead a n-class LM that can provide a probability to this OOV word based on its class. Consequently, in order to associate an accurate linguistic probability with each word, we combine a n-gram LM and a n-class LM, as inspired by [28]. We call this model a *word-to-class backoff LM*, because the n-class LM is used instead of the n-gram LM when the current word w_i, or at least one word of its history, is an *unknown word* (for example, an OOV word or an *absent substitution error*) and so does not belong to the vocabulary associated with the LMs. The probability of a word w_i is then given by Eq. 12:

$$P_{wc}(w_i|w_{i-n+1}^i) = \begin{cases} P_w(w_i|w_{i-n+1}^i) & \text{if } w_{i-n+1}^i \in V^n \\ P_c(w_i|w_{i-n+1}^i) & \text{otherwise} \end{cases} \quad (12)$$

with V being the vocabulary associated with the language models, $P_w(w_i|w_{i-n+1}^i)$ the probability given by the n-gram LM and $P_c(w_i|w_{i-n+1}^i)$ the probability given by the n-class LM.

To create the word-to-class backoff language model, the n-gram and the n-class LMs are built separately, using the classic maximum likelihood estimation (MLE). When building the n-class LM, classes are estimated for in-vocabulary words and for OOV words. However, the classes associated with *absent substitution errors* are not known. So, we consider that the set of classes associated with absent substitution errors is the whole set of classes of the LM; indeed, an *absent substitution error* can correspond to any word and thus to any class.

7.3.2 Use of the word-to-class backoff language model to correct errors

The aim of the correction step is to correct recognition errors on the top-list phrase, using a language model and competing recognition results in the word graph (for the *substitution errors* and *segmentation errors*; however, for the latter, the correct word is not certain to be one of the competing results). The language model used for that step is the word-to-class backoff LM, presented in the previous sub-section: the n-gram part of the backoff LM will be used on most of the words, except when the considered word (or one word in its history) is an OOV word or has been characterized as an *absent substitution error*: in that case, the n-class part of the LM is used instead. Fig. 6 shows a pruned word graph, where the part of the word-to-class backoff LM that is used
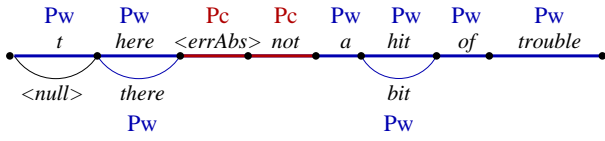
Fig. 6 Parts of the word-to-class backoff LM used to estimate the linguistic probability of each word of the word graph (Pw stands for the n-gram LM, while Pc stands for the n-class LM)

Fig. 7 Examples of phrases from the databases (a) ImadocSen-OnDB and (b) IAM-OnDB

to compute the linguistic probability of a word is given for each word of the top-list phrase. It can be seen that the n-class part of the LM is used to compute the linguistic probability for the word *"war"* (because it was detected as an absent substitution error), and for the word *"not"* (because the word in its history is an absent substitution error so it does not belong to the vocabulary). For the other words, the n-gram part of the word-to-class backoff LM can be used.

The corrected phrase finally corresponds to the best path in the pruned word graph, using Eq. 8 to combine the probability of words from the word-to-class backoff language model with their recognition probabilities previously given by the recognition system (*i.e.* provided in the N-best lists or computed as an approximation, as presented in Sect. 5.2).

8 Experiments and results

In this section, we report the experiments that were conducted to evaluate the whole error handling framework we have proposed. First, we describe the experimental setup, including a description of the handwritten and the linguistic data, as well as the two on-line handwritten phrase recognition systems we use to generate the N-best lists of phrases given as inputs to our framework. Then, we discuss the optimization of the parameters of the various parts of the framework (using validation sets). Finally, the overall results on the test set are presented.

8.1 Experimental setup

In this sub-section, we describe the linguistic data (LMs and associated vocabularies) and the handwritten databases we use in our experiments.

8.1.1 Handwritten data

The experiments were performed on two on-line handwritten phrase databases (Fig. 7 shows examples of phrases from the two databases). These databases are divided into different sets (at least one training set and one test set, and possibly validation sets) with no writer appearing in more than one dataset (writer-independent tasks are thus considered).

The two following paragraphs give greater details on these databases.

ImadocSen-OnDB[1] is an in-house database containing sentences acquired from a TabletPC. The written sentences have been extracted from part of the Brown corpus [16]. This is a simplified database since the sentences only contain lowercase letters (*i.e.* 26 different characters). Because it is of medium size, the database is divided into only two datasets: the training set, which is used both to train the classifiers (SVMs for the characterization step) and to optimize the parameters, and the test set, which is used to measure the final recognition results. Table 1 sums up the characteristics of the database. In this table, a *token* corresponds to a sequence of letters, a sequence of digits or a symbol (*e.g.* a punctuation mark).

Table 1 Characteristics of the handwritten sentence database ImadocSen-OnDB

Set	Train.	Test
# writers	25	17
# phrases	557	460
# tokens	8,769	7,080

IAM-OnDB [25] is a large database containing on-line handwritten texts acquired from a whiteboard. The written texts have been extracted from part of the LOB corpus [22] and contain 81 distinct characters (all lowercase and capital letters, punctuation marks, digits, a character for garbage symbols, and a character for the space). The IAM-OnDB-t2 task is considered here, which handles the recognition of handwritten lines. This database is divided into four datasets (as given in Table 2): the training set is used to train the classifiers, the two validation sets are used to optimize the other parameters, and the test set is used to measure the final recognition results using the whole framework.

---

[1] the ImadocSen-OnDB database can be downloaded at http://www.irisa.fr/imadoc/database/ImadocSen-OnDB.html

Table 2 Characteristics of the handwritten text database IAM-OnDB

Set	Train.	Test	Valid. 1	Valid. 2
# writers	97	68	24	17
# lines	5,364	3,859	1,438	1,518
# phrases	775	192	216	544
# tokens	35,166	24,542	8,595	9,670

8.1.2 Linguistic data

The language models used in the different parts of the framework are built on the tagged Brown corpus [16] using the SRILM toolkit [38]. This corpus contains 52,954 sentences (1,002,675 words), where 46,836 sentences (900,108 words) were used to train the LMs (we call this part of the corpus the *Brown training corpus*). The remaining sentences are not considered, because they are used in the Imadoc database (see Sect. 8.1.1). This ensures that the training set of the LMs and the handwritten test sets are, and remain, independent. We restricted the vocabulary associated with the LMs to the 20,000 words of the corpus that occur most frequently (other words in the corpus are considered as OOV words and are mapped to the tag <unk>). To train the probabilities of the LMs, the words of the corpus are divided into tokens (corresponding to the tokens previously defined).

The same LMs are used in the experiments on both databases. A bigram LM is used is used in the various steps of the framework: in the error detection part (used in the computation of posterior probabilities), in the error characterization part (to compute different word features), and in the error correction step (to be used as the main component of the word-to-class backoff LM). A 4-class LM is also used in the correction step (this time to be used as the backoff component of the word-to-class backoff LM). For this n-class LM, we consider 145 classes, which correspond to the POS classes in the tagged version of the Brown corpus.

8.2 Evaluation metrics

To evaluate the different steps of the error handling framework, we use different metrics. We present them in this subsection.

To evaluate the performance of the task, from a recognition point of view, we use the common *word accuracy* (WA) and *word recognition rate* (WRR), which are defined by

$$WA = \frac{\#words - \#subst - \#del - \#ins}{\#words} \quad (13)$$

and

$$WRR = \frac{\#words - \#subst - \#del}{\#words} \quad (14)$$

with $\#words$ being the number of all the words of the phrases to be recognized in the set considered and $\#subst$, $\#ins$,

and $\#del$ the number of substitutions, insertions, and deletions in the resulting recognized phrases, respectively.

To evaluate the posterior probabilities used as confidence indices, we compute the *normalized cross entropy* (NCE), which is commonly used to measure the quality of confidence measures (the higher its value, the better the confidence measure). The NCE is defined by:

$$NCE = \frac{H_{max} - H_{conf}}{H_{max}} \quad (15)$$

with

$$H_{max} = -p_c \log_2(p_c) - (1 - p_c) \log_2(1 - p_c) \quad (16)$$

and

$$H_{conf} = -\frac{1}{N} \left[\sum_{w_i \in W_{corr}} \log_2(p_i) + \sum_{w_i \in W_{err}} \log_2(1 - p_i) \right] (17)$$

with $p_c = \frac{N_c}{N}$ being the average probability that a word is correct (N_c is the number of correct words in the set considered, and N is the total number of words), p_i the predicted confidence that the word w_i is correct (given by the confidence measure), and W_{corr} and W_{err} the sets of correct words and error words, respectively.

The *Classification Error Rate* (CER) is used to measure the performance of the error detection step (and to chose the error threshold τ_{err}) and is defined as follows:

$$CER = \frac{\#correct_{err} + \#errors_{corr}}{\#words} \quad (18)$$

with $\#correct_{err}$ being the number of correct words identified as error hypotheses, and $\#errors_{corr}$ the number of errors identified as correct words.

Finally, the *precision* (Prec), the *recall* (Rec), and the *F-Measure* (F) are classical measures used in the field of information retrieval. Here, they are used to assess the performance of identifying errors. They are defined as follows:

$$Prec = \frac{\#errors_{err}}{\#errors_{err} + \#errors_{corr}}, \quad (19)$$

$$Rec = \frac{\#errors_{err}}{\#errors_{err} + \#correct_{err}}, \quad (20)$$

and

$$F = \frac{2 \times Prec \times Rec}{Prec + Rec} \quad (21)$$

with $\#errors_{err}$ being the number of error hypotheses correctly identified as error hypotheses. Prec gives the percentage of word classified as errors and that are indeed errors, whereas Rec gives the percentage of errors to identify that are indeed classified as errors. The F-measure is then a single-valued metric that reflects the trade-off between the precision and the recall.

8.3 Baseline phrase recognition systems

We used two on-line handwriting recognizers in our experiments. In this sub-section, we present both, as well as the baseline recognition results on the two datasets (*i.e.* the recognition results before any error handling approach is used).

8.3.1 ResifPhrase sentence recognition system

The first recognizer we used is the on-line handwritten sentence recognition system RESIFPhrase [33]. Given an input handwritten sentence, a graph containing handwritten word segmentation hypotheses is built. To identify these hypotheses, a radial basis function network (RBFN) is used to classify each inter-stroke gap. A confidence index is associated with each of these classification results and is used to create additional segmentation hypotheses. A MAP decoding is then performed on the word graph to find the likeliest sentence, using graphical and linguistic information as given by Eq. 7. In this case, $P(S|W_k)$ is, in fact, the accumulated score given by the word recognition system: it combines graphic and lexical scores given by the word recognition system RESIFMot [6]. The *graphic score* includes adequation measures between each character and its corresponding model, as well as spatial and statistical information between characters; the *lexical score* depends on edit operations performed during the lexical post-processing step. $P(W_k)$ is given by a bigram LM trained on the same part of the Brown corpus as the LMs used in the error handling framework. Nevertheless, it is different from these LMs because its vocabulary contains 13,748 words made up of lowercase letters only (the same vocabulary is used in the RESIFMot recognizer).

The limitation of this recognizer is that it can only recognize lowercase letters which restricts its use to ImadocSen-OnDB. For this reason, we consider another recognizer to evaluate our error handling framework on the IAM-OnDB, which is a more realistic database.

8.3.2 Microsoft text recognizer

To obtain N-best lists of phrases from on-line handwritten texts (as given in IAM-OnDB), we used the recognizer provided by Microsoft in their TabletPC sdk [31]. This enables us to evaluate our whole approach with a real "black box" recognizer, since the N-best lists only contain phrase hypotheses (ordered according to a recognition score, but one that is not given). Furthermore, neither the vocabulary used in the recognizer nor the LMs involved are known.

8.3.3 Baseline recognition results

In the following experiments, we consider three error handling tasks, depending on the handwritten database considered and the recognition system used to provide the input N-best list of phrase hypotheses. The tasks are:

- Sen/RP: RESIFPhrase recognizer on ImadocSen-OnDB;
- Sen/MS: Microsoft recognizer on ImadocSen-OnDB;
- Txt/MS: Microsoft recognizer on IAM-OnDB.

Table 3 gives the baseline WA and WRR (defined in Sect. 8.2), for each task.

Table 3 Baseline recognition results on the test sets

Task	Sen/RP	Sen/MS	Txt/MS
Top-1 WA	83.1 %	94.2 %	85.2 %
Top-1 WRR	84.6 %	94.7 %	86.7 %
Top-N WRR	90.3 %	95.5 %	92.0 %

The WA and the WRR are first given, considering only the top-1 phrase on the list: these are the baseline rates against which those obtained with our error handling approaches will be compared. The recognition rate when considering the top-N phrases of the list is also given (here, N is set to 150): it gives an upper bound for our approach. Indeed, it can only be achieved if all the word error hypotheses are perfectly detected and perfectly corrected (the correction step only uses words of the competing phrases to correct the errors). It also implies a perfect characterization step, since the correction step is performed on the graph pruned according to the characterized error types. Thus, we can see that the WRR for the first and the third tasks could be improved (due to about a 6 % difference between the top-1 and top-N rates), whereas it might be more difficult to improve the WRR for the second task (only a 1 % difference between the top-1 and top-N rates).

The rates obtained with the Microsoft recognizer are higher than those obtained with the RESIFPhrase recognizer. However, these results cannot be compared directly, since the Microsoft recognizer uses a larger vocabulary, more sophisticated LMs, and is also trained on a much larger training set. Furthermore, the rates with the Microsoft recognizer are higher than those obtained in [20] (using also the Microsoft recognizer on IAM-OnDB), which is due to the fact that, here, we compute rates on the tokens and not on the words. Thus, a handwritten compound word corresponding to 3 tokens (the first word, the dash, and the second word) will only be recognized for [20] if all its tokens are recognized, whereas we will focus here on how many of its tokens are recognized.

8.4 Comparison of confidence measures to detect error hypotheses

In the experiments presented in this sub-section, we compare the confidence measures used to detect word error hy-

potheses. Table 4 gives the NCE for the word posterior probabilities, and for the two word posterior probability approximations presented in Sect. 5.2 (the NCE is computed on the training set for ImadocSen-OnDB, and on the first validation set for IAM-OnDB).

**Table 4** Quality of the posterior probabilities using the NCE

Task	Sen/RP	Sen/MS	Txt/MS
$P_{post}(w_i)$	0.275	undef.	undef.
$\widetilde{P}_{post}(w_i)$ (using $f_{rel}(w_i)$)	0.253	0.150	0.193
$\widetilde{P}_{post}(w_i)$ (using $f_{rank}(w_i)$)	0.258	0.154	0.197

We can see that the posterior probability obtains the best NCE value, but the approximated posterior probabilities based on the relative frequency and on the rank frequency also give good values. The rank frequency gives better results than just using the frequency because it gives a lower score to a word that appears only in the last phrases of the list than to a word that appears the same number of times in the first phrases. So, for the Sen/MS and Txt/MS tasks, we choose to use the rank-weighted frequency of words to approximate the word recognition probabilities, so that we can compute approximated posterior probabilities for the words (because no word recognition probability is provided by the Microsoft recognizer), while we use the actual recognition probabilities for Sen/RP.

8.5 Setting of the error detection threshold

In this sub-section, we present the experiments on the choice of the error detection threshold, now that we have chosen the word posterior probability approximations (see Sect. 8.4). In Fig. 8, ROC curves are given, when using the word posterior probabilities as confidence indices (see Sect. 5.3): each point of these curves corresponds to a chosen threshold τ_{err} and shows the compromise between the correct words, the confidence score of which is above τ_{err} (TAR, for *True Acceptance Rate*), and the error words, the confidence score of which is above τ_{err} (FAR, for *False Acceptance Rate*). To plot these curves, we used the same training and validation sets as previously, for the three tasks.

For the tasks Sen/MS and Txt/MS, it can be seen that ROC curves are almost lines from $(0\%, 0\%)$ to $(100\%, 100\%)$, which means that error words cannot really be separated from correct words, at least using only their posterior probabilities. This may be because of the absence of a word recognition score given by the recognition system: the rank frequency score we use may not be good enough to compute the approximated posterior probabilities. In order not to detect too many false errors (*i.e.* that are indeed correct words), we set the threshold to 0.1. For the Sen/RP task, the ROC
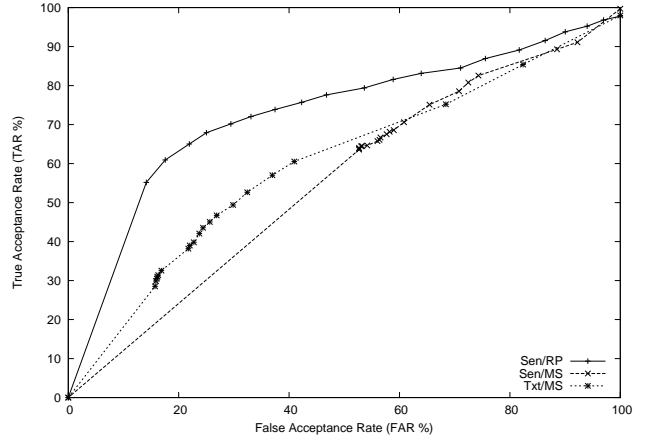
Fig. 8 ROC curve comparing confidence measures for the different error handling tasks

curve obtained has a better shape and allows us to set the error detection threshold to 0.25 (corresponding to a 88.5% TAR and a 79.8% FAR).

8.6 Choice of the feature set to characterize error hypotheses

In the experiments described in this sub-section, we compare the various feature sets used in a SVM to characterize word error hypotheses into the three predefined error types (see Sect. 6).

A SVM is trained for each of the three tasks using parameter optimization with a 10-fold cross-validation, thanks to the LIBSVM library [7]. The training sets of the two databases are used to train the SVMs; these training sets are limited to the actual error words (the correct words are discarded from the sets).

Table 5 gives the accuracy, that corresponds to the word recognition rate WRR (here, computed only on the error words), for each feature set considered and each task (it is the metric used in LIBSVM to optimize the parameters of the SVMs trained).

Table 5 Accuracy of the SVMs for the various feature sets

Task	Sen/RP	Sen/MS	Txt/MS
wSet	58.5 %	77.2 %	58.7 %
lSet	58.6 %	77.7 %	58.8 %
nSet	61.8 %	79.7 %	60.6 %

The WRR obtained using the *wSet* are already good and are only slightly improved when using the next context, *lSet*. The improvement is more significant when using the larger context, *nSet*, which improves the WRR by 2 to 3 %.

In order to study the impact of the different features, we used a tool from LIBSVM to perform feature selection: it computes the contribution of each feature, in terms of the F-measure Fand then computes the WRR for classifiers trained with different subsets of the features. The better WRR is thus obtained when using the whole set of features, *nSet*. Table 6 also shows the importance of each feature, considering the three previous tasks (1 stands for the most important feature and 16 for the less important one).

Table 6 Importance of the various features, for the characterization classifiers

Task	Sen/RP	Sen/MS	Txt/MS
wWordPosteriorProba	6	4	6
wWordUnigramProba	13	10	11
wWordLength	5	8	7
wWordPhrasePos	10	7	4
wPhraseLength	9	9	5
lConcurrNbWords	12	2	2
lConcurrNullEdge	2	5	10
lConcurrPosteriorProbasMean	3	3	3
lConcurrPosteriorProbasVariance	15	14	15
lConcurrUnigramProbasMean	8	16	13
lConcurrUnigramProbasVariance	11	6	12
nWordBigramProba	14	11	9
nPrevWordPosteriorProba	4	13	8
nNextWordPosteriorProba	16	15	16
nPrevWordError	1	1	1
nNextWordError	7	12	14

It can be seen that the most important information to characterize error hypotheses into the predefined types are:

– features from *wSet* (minus the unigram probability feature);
– the mean of posterior probabilities of concurrent words as well as their number and the presence of a *null* edge among them;
– information on the previous word of the top-list phrase (if detected as an error, and its posterior probability).

This explains the good results obtained when using only *wSet* and the improvement brought by information on the previous word of the top-list phrase.

8.7 Evaluation of the overall error handling approach

In this sub-section, we present the final results on the test sets of both databases, using the parameters optimized in the previous sub-sections, for the three error handling tasks. Table 7 gives the word recognition rate WRR, and the classification error rate CER, for the three different tasks, using the whole error handling approach (see Sect. 8.2 for their definitions). Moreover, to measure more precisely the performance of the error handling process, the precision (Prec), the recall (Rec), and the F-Measure (F) are given in Table 8.

The WRR is decreased for the three tasks, when using only the error detection step (by 9-10 %). Consequently, the CER is increased by $\sim$10 % for the Sen/MS and Txt/MS tasks but only by $\sim$5 % for the Sen/RP task. When adding the correction step, the WRR is improved, for the three tasks, when compared to the error detection step alone, but it is still below the baseline rates (by $\sim$5 %). Likewise, the CER is decreased (and is $\sim$3-5 % over the baseline CER). This behaviour is unavoidable because correct words are selected during the error detection step, and not all of the so-detected errors can be corrected during the correction step.

As this work is not placed in the context of a recognition task, we evaluate more precisely the error handling contribution, using classic methods from the field of information retrieval (see Table 8). It can be seen that the recall is decreased during the correction process because of the attempt to correct detected error hypotheses, but the correction step also improves the precision and the F-measure. More particularly, better results are obtained for the Sen/RP task, where the actual recognition scores of the words are used during the correction step, to retrieve the final phrase.

9 Conclusion

We have proposed a framework to handle recognition errors on phrase transcripts, from handwriting recognition systems, that are meant to be used as inputs to a higher-level system (*e.g.* information retrieval systems or text categorization systems). The framework takes a N-best best list of phrases (given by the recognizer) as input and outputs a phrase on which errors are detected, characterized, and even corrected. This approach is decomposed into four steps: an alignment step (where a word graph is built by aligning the N-best phrases of the list), a detection step (where word posterior probabilities are used as confidence measures to detect error hypotheses on the words in the top-list phrase), a characterization step (where the previously detected error hypotheses are characterized into predefined error types, which are also used to prune the word graph accordingly), and a correction step (where a word-to-class backoff LM is defined and used to retrieve the final phrase on the pruned graph and thus to correct initial recognition errors). Experiments on two handwritten phrase databases were performed, using two recognition systems to provide the initial N-best lists: thus, three tasks were considered (Sen/RP, Sen/MS, and Microsoft). The results of this first implementation are mitigated. Indeed, even though the features chosen were proven to allow the characterization of error hypotheses into predefined error types and the correction step was shown to be

Table 7 Overall recognition results with the whole error handling approach

| Task | Sen/RP | | Sen/MS | | Txt/MS | |
Rate	WRR	CER	WRR	CER	WRR	CER
Baseline	84.6 %	15.2 %	94.7 %	5.3 %	86.70 %	13.0 %
Error detect.	76.1 %	20.8 %	85.0 %	14.2 %	75.3 %	22.7 %
Error detect. & charact. & correct.	79.7 %	18.2 %	89.4 %	10.1 %	80.2 %	17.1 %

Table 8 Precision and recall results with the whole error handling approach

| Task | Sen/RP | | | Sen/MS | | | Txt/MS | | |
Rate	Prec	Rec	F	Prec	Rec	F	Prec	Rec	F
Error detect.	20.2 %	21.8 %	20.9 %	2.7 %	15.0%	9.7 %	11.3 %	14.9 %	12.8 %
Error detect. & charact. & correct.	29.2 %	17.5 %	21.9 %	12.8 %	13.3 %	13.0 %	17.7 %	11.3 %	13.8 %

an added-value to the error detection step alone, the overall results (in terms of WRR) are below the baseline WRR. Starting from the results obtained in terms of precision and recall, further investigations will be needed to improve the different steps of the proposed error handling framework.

Future works will investigate approaches to align multiple segmentations to better correct segmentation errors. To do so, we could align multiple segmentation hypotheses when creating the word graph, as proposed in [43], where an edge can be align with two other edges, when the graph is created. We will study the bias of the posterior probabilities, as it was shown in [21] to impact the NCE and thus the quality of confidence measures based on these probabilities. We will investigate using other features for the characterization step and/or the detection step, especially probabilities given by n-class LMs using POS classes (as the n-class part of our word-to-class backoff LM) that were proven to be efficient, in recent works in speech recognition [36, 13]. It would also be of interest to investigate using other measures to optimize the error detection threshold, like the F-measure, and to study how it impacts the correction step and thus the overall error handling process. We will also investigate how to correct words, not only using the words that appear in the input phrase list. Indeed, it would be interesting to handle words characterized as *absent substitution errors*, for which the correct recognition result does not appear in the word graph. In that case, we could use their classes, as identified by the word-to-class backoff LM, during the error correction step. The class of a word may then be used to select a specific vocabulary, containing only words of that class. This specific vocabulary could be given to a word recognition system, which could be used to further recognize the word. Finally, it would be of interest to measure how the error handling approach impacts the performance of higher-level applications that deal with handwritten transcripts (for information retrieval or categorization tasks, for example), as in [26], or to study how to present all these error charac-

terization and correction results to an end-user (in a freeform note-taking application, for example).

References

1. Abdou, S., Scordilis, M.: Beam search pruning in speech recognition using a posterior probability-based confidence measure. Speech Communication **42**(3-4), 409–428 (2004)
2. Bertolami, R., Bunke, H.: Integration of n-gram language models in multiple classifier systems for offline handwritten text line recognition. International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI) **22**(7), 1301–1321 (2008)
3. Bertolami, R., Zimmermann, M., Bunke, H.: Rejection strategies for offline handwritten text recognition. Pattern Recognition Letters **27**, 2005–2012 (2006)
4. Bisani, M., Ney, H.: Open vocabulary speech recognition with flat hybrid models. In: Proc. of the European Conference on Speech Communication and Technology (Eurospeech), pp. 725–728. Lisbon, Portugal (2005)
5. Blatz, J., Fitzgerald, E., Foster, G., Gandrabur, S., Goutte, C., Kulesza, A., Sanchis, A., Ueffing, N.: Confidence estimation for machine translation. In: Proc. of the International Conference on Computational Linguistics (COLING), pp. 315–321. Geneva, Switzerland (2004)
6. Carbonnel, S., Anquetil, E.: Lexicon organization and string edit distance learning for lexical post-processing in handwriting recognition. In: Proc. of the International Workshop on Frontiers in Handwriting Recognition (IWFHR), pp. 462–467. Tokyo, Japan (2004)
7. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines (2001). Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm
8. Chase, L.: Word and acoustic confidence annotation for large vocabulary speech recognition. In: Proc. of the European Conference on Speech Communication and Technology (Eurospeech), pp. 815–818. Rhodes, Greece (1997)
9. Cox, S., Dasmahapatra, S.: High-level approaches to confidence estimation in speech recognition. IEEE Transactions on Speech and Audio Processing **10**(7), 460–471 (2002)
10. Evermann, G., Woodland, P.: Posterior probability decoding, confidence estimation and system combination. In: Speech Transcription Workshop. College Park, United States (2000)

11. Falavigna, D., Gretter, R., Riccardi, G.: Acoustic and word lattice based algorithms for confidence scores. In: International Conference on Spoken Language Processing, pp. 1621–1624. Denver, United States (2002)

12. Farooq, F., Jose, D., Govindaraju, V.: Phrase-based correction model for improving handwriting recognition accuracies. Pattern Recognition **42**(12), 3271–3277 (2009)

13. Fayolle, J., Moreau, F., Raymond, C., Gravier, G.: Reshaping automatic speech transcripts for robust high-level spoken document analysis. In: Proc. of the Workshop on Analytics for Noisy Unstructured Text Data (AND). Toronto, Canada (2010)

14. Fiscus, J.: A post-processing system to yield reduced word error rates: Recognizer output voting error reduction. In: Proc. of the Workshop on Automatic Speech Recognition and Understanding (ASRU), pp. 347–352. Santa Barbara, United States (1997)

15. Forney, G.: The viterbi algorithm. IEEE **61**(3), 268–278 (1973)

16. Francis, W., Kucera, H.: Brown Corpus Manual. Brown University, Providence, United States (1979)

17. Fu, Y., Du, L.: Combination of multiple predictors to improve confidence measure based on local posterior probabilities. In: Proc. of the International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 93–96. Philadelphia, United States (2005)

18. Gandrabur, S., Foster, G., Lapalme, G.: Confidence estimation for nlp applications. ACM Transactions on Speech and Language Processing **3**(3), 1–29 (2006)

19. Goodman, J.: A Bit of Progress in Language Modeling. Technical report MSR-TR-2001-72, Microsoft Research, Redmond, United States (2001)

20. Graves, A., Liwicki, M., Fernandez, S., Bertolami, R., Bunke, H., Schmidhuber, J.: A novel connectionist system for unconstrained handwriting recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) **31**(5), 855–868 (2009)

21. Hillard, D., Ostendorf, M.: Compensating for word posterior estimation bias in confusion networks. In: Proc. of the International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1153–1156. Toulouse, France (2006)

22. Johansson, S., Leech, G., Goodluck, H.: Manual of Information to accompany the Lancaster-Oslo/Bergen Corpus of British English, for use with digital Computers. University of Oslo, Oslo, Norway (1978)

23. Kemp, T., Schaaf, T.: Estimating confidence using word lattices. In: Proc. of the European Conference on Speech Communication and Technology (Europseech), pp. 827–830. Rhodes, Greece (1997)

24. Levenshtein, V.: Binary codes capable of correcting deletions, insertions and reversals. Soviet Physics Doklady **10**(8), 707–710 (1966)

25. Liwicki, M., Bunke, H.: IAM-OnDB – an on-line English sentence database acquired from handwritten text on a whiteboard. In: Proc. of the International Conference on Document Analysis and Recognition (ICDAR), pp. 956–961. Seoul, Korea (2005)

26. Lopresti, D.: Optical character recognition errors and their effects on natural language processing. International Journal on Document Analysis and Recognition (IJDAR) **12**(3), 141–151 (2009)

27. Marukatat, S., Artieres, T., Gallinari, P.: Rejection measures for handwriting sentence recognition. In: Proc. of the International Workshop on Frontiers in Handwriting Recognition (IWFHR), pp. 24–29. Ontario, Canada (2002)

28. Niesler, T.: Category-based statistical language models. Ph.D. thesis, University of Cambridge, Cambridge, United Kingdom (1997)

29. Perraud, F., Viard-Gaudin, C., Morin, E., Lallican, P.M.: Statistical language models for on-line handwriting recognition. IEICE Transactions on Information and Systems **E88-D**(8), 1807–1814 (2005)

30. Pitrelli, J., Subrahmonia, J., Perrone, M.: Confidence modeling for handwriting recognition: algorithms and applications. International Journal on Document Analysis and Recognition (IJDAR) **8**(1), 35–46 (2006)

31. Pittman, J.: Handwriting recognition: Tablet PC text input. IEEE Computer **40**(9), 49–54 (2007)

32. Quiniou, S., Anquetil, E.: A priori and a posteriori integration and combination of language models in an on-line handwritten sentence recognition system. In: Proc. of the International Workshop on Frontiers in Handwriting Recognition (IWFHR), pp. 403–408. La Baule, France (2006)

33. Quiniou, S., Bouteruche, F., Anquetil, E.: Word extraction associated with a confidence index for on-line handwritten sentence recognition. International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI) **23**(5), 945–966 (2009)

34. Quiniou, S., Cheriet, M., Anquetil, E.: Handling out-of-vocabulary words and recognition errors based on word linguistic context for handwritten sentence recognition. In: Proc. of the International Conference on Document Analysis and Recognition (ICDAR), pp. 466–470. Barcelona, Spain (2009)

35. Saldarriaga, S.P., Viard-Gaudin, C., Morin, E.: Impact of on-line handwriting recognition performance on text categorization. International Journal on Document Analysis and Recognition (IJDAR) **13**(2), 159–171 (2010)

36. Shi, Y., Zhou, L.: Error detection using linguistic features. In: Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 41–48. Vancouver, Canada (2005)

37. Siu, M., Gish, H.: Evaluation of word confidence for speech recognition systems. Computer Speech and Language **13**(4), 299–319 (1999)

38. Stolcke, A.: Srilm – an extensible language modeling toolkit. In: Proc. of the International Conference on Spoken Language Processing (ICSLP), pp. 901–904. Denver, United States (2002). Available at http://www.speech.sri.com/projects/srilm/

39. Subramaniam, L., Roy, S., Faruquie, T., Negi, S.: A survey of types of text noise and techniques to handle noisy text. In: Proc. of the Workshop on Analytics for Noisy Unstructured Text Data (AND), pp. 115–122. Barcelona, Spain (2009)

40. Ueffing, N., Ney, H.: Word-level confidence estimation for machine translation. Computational Linguistics **33**(1), 9–40 (2007)

41. Wessel, F., Schlter, R., Macherey, K., Ney, H.: Confidence measures for large vocabulary continuous speech recognition. IEEE Transactions on Speech and Audio Processing **9**(3), 288–298 (2001)

42. Williams, G., Renals, S.: Confidence measures from local posterior probability estimates. Computer Speech and Language **13**(4), 395–411 (1999)

43. Xue, J., Zhao, Y.: Improved confusion network algorithm and shortest path search from word lattice. In: Proc. of the International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 853–856. Philadelphia, United States (2005)

44. Zhou, Z.Y., Meng, H.: A two-level schema for detecting recognition errors. In: International Conference on Spoken Language Processing (ICSLP), pp. 449–452. Jeju Island, Korea (2004)

45. Zimmermann, M., Chappelier, J.C., Bunke, H.: Offline grammar-based recognition of handwritten sentences. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) **28**(5), 818–821 (2006)