On the Optimal Decision Rule for Sequential Interactive Structured Prediction

Vicent Alabau*, Alberto Sanchis, Francisco Casacuberta

Institut Tecnològic d'Informatica, Universitat Politècnica de València, Camino de Vera s/n, 46022 València, Spain

---

Abstract

Interactive structured prediction (ISP) is an emerging framework for *structured prediction* (SP) where the user and the system collaborate to produce a *high quality* output. Typically, search algorithms applied to ISP problems have been based on the algorithms for fully-automatic SP systems. However, the decision rule applied should not be considered as optimal since the goal in ISP is to reduce human effort instead of output errors. In this work, we present some insight into the theory of the sequential ISP search problem. First, it is formulated as a *decision theory* problem from which a general analytical formulation of the optimal decision rule is derived. Then, it is compared with the standard formulation to establish under what conditions the standard algorithm should perform similarly to the optimal decision rule. Finally, a general and practical implementation is given and evaluated against three classical ISP problems: interactive machine translation, interactive handwritten text recognition, and interactive speech recognition.

Keywords: Interactive pattern recognition, Minimum Bayes risk, Human interaction, Machine translation, Handwritten text recognition, Automatic speech recognition

---

1. Introduction

Structured prediction[1] (SP) [9] is a classification problem in which the output consists of structured labels (as opposed to independent labels), i.e., the output labels have dependencies on each other. Examples of structured outputs are natural language, DNA sequences, or an XML describing the layout analysis of a web page. Traditionally, SP has been approached as a fully automated procedure in which an input ($\boldsymbol{x}$) is presented to a SP system and the SP system produces an output ($\hat{\boldsymbol{y}}$). As a post-processing, it is advisable for a human expert to revise the system output to amend the possible errors in order to produce the final output (or reference), $\boldsymbol{r}$. This *post-editing* (PE) process (depicted in Figure 1(a)) is completely manual and all the supervision effort is delegated to the user.

The *interactive structured prediction* (ISP) framework (also known as *interactive pattern recognition* [12]) was introduced in [13] to reduce the cost of correcting the automatically generated output. In ISP, the user is introduced in the core of a SP system so that the system and the user can interact with each other to minimize the effort required to produce a satisfactory output (Figure 1(b) represents the ISP interaction scheme). In ISP, an input $\boldsymbol{x}$ is given to the system, which outputs a possible hypothesis $\hat{\boldsymbol{y}}$. Then, the user analyzes $\hat{\boldsymbol{y}}$ and provides feedback $\boldsymbol{f}$ regarding some of the errors committed. Now, the system can benefit from the feedback to propose a new improved hypothesis. This process is repeated until the user finds a satisfactory solution, $\boldsymbol{r}$, and the process ends.

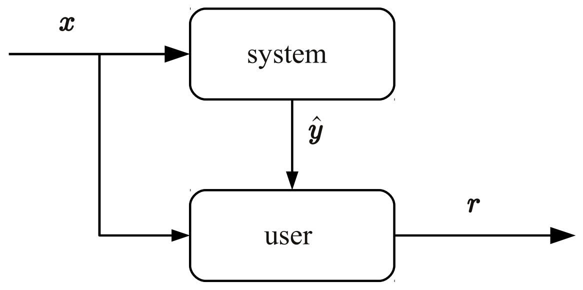
Traditionally, SP systems are designed following a decision rule to minimize output errors. However, this is not optimal for ISP since the decision rule should be formalized in terms of minimizing user interactions. Indeed, this fact was proved in [7], where an alternative strategy is applied to a specific case of ISP (i.e., text prediction). Inspired by that work, here we provide an optimal decision rule for ISP which covers a broader range of common ISP problems in which the output depends on a structured input $\boldsymbol{x}$. This strategy is analyzed from a theoretical perspective and a

---

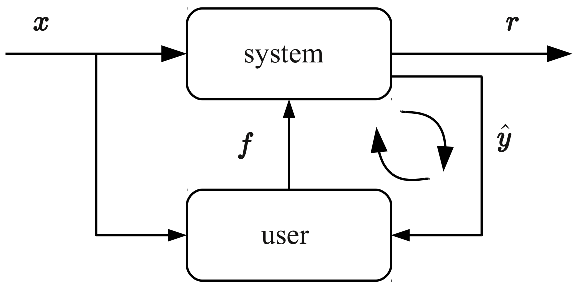
*Corresponding author: Phone: (34) 96 387 70 69, Fax: (34) 96 387 72 39.

Email addresses: `valabau@iti.upv.es` (Vicent Alabau), `asanchis@iti.upv.es` (Alberto Sanchis), `fcn@iti.upv.es` (Francisco Casacuberta)

[1]This was also known as syntactic or structural pattern recognition [4]



(a) Diagram of the *post-editing* process. The system processes the input $\boldsymbol{x}$ to produce an output $\hat{\boldsymbol{y}}$. Then, the user, who knows the desired output $\boldsymbol{r}$, modifies $\hat{\boldsymbol{y}}$ to create the final output $\boldsymbol{r}$.



(b) Diagram of the *interactive structured prediction* process. The system processes the input $\boldsymbol{x}$ to produce an initial output $\hat{\boldsymbol{y}}$. Then, the user analyzes $\hat{\boldsymbol{y}}$ and proposes a correction by some feedback $\boldsymbol{f}$. Now, the system proposes a new hypothesis $\hat{\boldsymbol{y}}$. This process is repeated until the desired solution $\boldsymbol{r}$ is obtained.

Figure 1: Diagram of the *post-editing* process (left) and the *interactive structured prediction* process (right).

practical decoding algorithm is developed to be used straightforwardly in many ISP tasks. In addition, we propose theoretically and empirically that the algorithm that has been used for ISP until now is a good approximation to the optimum for ISP.

The remainder of this article is organized as follows. First, the ISP framework is formalized in Section 2. Section 3 describes an optimal decision rule to minimize the number of interactions under the *decision theory* perspective. A general and practical algorithm is developed in Section 4. Finally, empirical results are shown in Section 5 and conclusions and future work are described in Section 6.

2. Sequential Interactive Structured Prediction

Sequential interactive structured prediction (SISP) is a specific case of ISP where the user validates and/or corrects the system output in sequential order (typically left-to-right). This is especially interesting for many *natural language processing* (NLP) tasks, since humans usually listen, read, write, and talk in sequential order.

Let $\boldsymbol{y}^{(i)} = \boldsymbol{p}^{(i)} \cdot \boldsymbol{s}^{(i)}$ be a hypothesis at iteration (i) that is a concatenation of a correct prefix $\boldsymbol{p}^{(i)}$ of the solution $\boldsymbol{r}$ and a suffix hypothesis $\boldsymbol{s}^{(i)}$. Then, the protocol that rules SISP to obtain $\boldsymbol{r}$ can be formulated in the following steps:

0. Initially ($i = 0$), the correct prefix is the empty string, $\boldsymbol{p}^{(0)} = \lambda$, and the system proposes a complete hypothesis $\hat{\boldsymbol{s}}^{(0)}$.
1. At iteration $(i \geq 1)$, the user finds the longest prefix $\boldsymbol{a}^{(i)}$ of $\hat{\boldsymbol{s}}^{(i-1)}$ that is error-free and corrects the first error in the suffix, which, let us assume, is at position k, with r_k.
2. A new extended prefix $\boldsymbol{p}^{(i)}$ is produced as a concatenation $\boldsymbol{p}^{(i-1)} \cdot \boldsymbol{a}^{(i)}$ and the new introduced word r_k.
3. Then, the system proposes a suffix hypothesis $\hat{\boldsymbol{s}}^{(i)}$ that follows the prefix $\boldsymbol{p}^{(i)}$ established in the previous step.
4. Steps 1, 2 and 3 are iterated until, at some iteration $i = I$ with $I \leq |\boldsymbol{r}|$, a correct solution is obtained, $\hat{\boldsymbol{y}}^{(I)} \equiv \boldsymbol{r}$.

Figure 2 shows an example of this protocol for SISP. Note that, when the user introduces r_4 =‘*is*’, the system amends automatically the word ‘*cannot*’. Similarly, introducing r_7 =‘*in*’ corrects ‘*web*’. Thus, in a PE system, the user would have needed to make five corrections, whereas just two corrections were needed in SIPS.

3. Optimal Decision Rule for SISP

Following the *minimum classification error* (MCE) [2], the optimum decision rule is the one that minimizes the average probability of loss (conditional risk) over a probability distribution $Pr(\cdot)$. In classification problems, the human effort can be approximated by a cost 0 if the output is correct and a cost 1 if the user has to amend an

SOURCE ($\boldsymbol{x}$):	si alguna función no se encuentra disponible en su red
REFERENCE ($\boldsymbol{r}$):	if any feature is not available in your network

$i = 0$	$\boldsymbol{p}^{(0)}$	
	$\hat{\boldsymbol{s}}^{(0)}$	if any feature cannot be found on your web
	$\hat{\boldsymbol{y}}^{(0)}$	if any feature cannot be found on your web
$i = 1$	$\boldsymbol{a}^{(1)}$	*if any feature*
	r_4	**is**
	$\boldsymbol{p}^{(1)}$	if any feature is
	$\hat{\boldsymbol{s}}^{(1)}$	not available at your web
	$\hat{\boldsymbol{y}}^{(1)}$	if any feature is not available at your web
$i = 2$	$\boldsymbol{a}^{(2)}$	*not available*
	r_7	**in**
	$\boldsymbol{p}^{(2)}$	if any feature is not available in
	$\hat{\boldsymbol{s}}^{(2)}$	your network
	$\hat{\boldsymbol{y}}^{(2)}$	if any feature is not available in your network
$I = 2$	$\boldsymbol{y}^{(2)} \equiv \boldsymbol{r}$	if any feature is not available in your network

Figure 2: Example of an ISP session for a machine translation task from Spanish to English. The source sentence is the input $\boldsymbol{x}$ while the reference $\boldsymbol{r}$ is the result that the user has in mind. At each iteration (i), $\hat{\boldsymbol{s}}^{(i)}$ is the suffix proposed by the system. $\boldsymbol{a}^{(i)}$ (in *italics*) is the longest correct prefix of $\hat{\boldsymbol{s}}^{(i-1)}$. Finally, r_k (in **boldface**) is the word introduced by the user to amend the error, which results in a new validated prefix $\boldsymbol{p}^{(i)}$.

erroneously classified sample by assigning the correct label. This function is known as the *zero-one* loss function and leads to the *maximum-a-posteriori* (MAP) decision rule [2]:

$$\hat{\boldsymbol{y}} = \underset{\boldsymbol{y}}{\operatorname{argmax}}\, Pr(\boldsymbol{y}|\boldsymbol{x}) \tag{1}$$

Typically, SISP systems have used an extension of MAP which has been successfully applied to several NLP tasks [12]. Thus, at steps 0 and 3 of the SIPS protocol, a suffix hypothesis $\hat{\boldsymbol{s}}^{(i)}$ that continues a validated prefix $\boldsymbol{p}^{(i)}$ is generated using the following equation:

$$\hat{\boldsymbol{s}}^{(i)} = \underset{\boldsymbol{s}}{\operatorname{argmax}}\, Pr(\boldsymbol{s}|\boldsymbol{x}, \boldsymbol{p}^{(i)}) \tag{2}$$

where the output in the interaction (i) is $\hat{\boldsymbol{y}}^{(i)} = \boldsymbol{p}^{(i)} \cdot \hat{\boldsymbol{s}}^{(i)}$.

Nevertheless, the MAP rule seeks solutions with zero errors, whereas we would prefer a hypothesis that minimizes the number of human corrections.

3.1. The Cost of Interactively Correcting the Output

Traditionally, the cost of post-editing a sentence is measured in terms of the edit distance (i.e., the minimum number of substitutions, insertions, and deletions needed to transform the system output into the reference). A specific case is the Hamming distance, where only substitutions are needed since there is a one-to-one correspondence between the input and the output. Of course, the edit distance is a simplistic approach to assess PE effort. On the one hand, it is optimistic regarding the number of operations to make, since a human will hardly ever perform the operations with the *minimum* number of operations, especially in complex problems. On the other hand, the edit distance assigns the same cost to all the operations, regardless of the complexity of the problem and the cognitive effort needed to perform them. On the positive side, the edit distance provides an automatic and intuitive measure. Consequently, it has been widely adopted for many NLP tasks as the PE cost.

Analogously, in ISP systems, the cost of interactively correcting a system output can be computed as the number of corrections (substitutions) needed to obtain the reference. Note that this is not equivalent to the Hamming distance since the part on the right of the output may change after each user correction. In this case, the cognitive effort is also neglected and the substitution cost is the same for all corrections. Besides, system suggestions may influence human corrections, since a good proposal could change a user's opinion regarding what the correct solution is. In this sense, using a unique reference can be deemed as a pessimistic approach in problems with multiple correct solutions. Notwithstanding, this criteria can be considered to be a reasonably valid approximation.

In the following, we will denote with y_k the k-th element in $\boldsymbol{y}$ and $y^{(i)}_{j..k-1}$ as the substring from j to $k-1$ of $\boldsymbol{y}^{(i)}$.

Proposition 1. *Let $Pr(\boldsymbol{s}|\boldsymbol{x},\mathbf{p}^{(i)})$ be a posterior probability over the suffixes that continue $\mathbf{p}^{(i)} = (y^{(i)}_1, \dots, y^{(i)}_{j-1})$. A suffix $\hat{\mathbf{s}}^{(i)} = (\hat{y}^{(i)}_j, \dots, \hat{y}^{(i)}_k, \dots, \hat{y}^{(i)}_{\hat{I}})$ with the last symbol at position $\hat{I}$, which minimizes the conditional risk of the number of interactions, can be obtained following this decision rule:*

$$\hat{y}^{(i)}_k = \operatorname*{argmax}_{y_k} \sum_{\boldsymbol{s}'} Pr(y_k \cdot \boldsymbol{s}'|\boldsymbol{x},\mathbf{p}^{(i)} \cdot \hat{y}^{(i)}_{j..k-1})$$
$$\textit{for } k = j \dots \hat{I} \wedge \hat{y}^{(i)}_{\hat{I}+1} \equiv \$ \qquad (3)$$

where $\boldsymbol{s}'$ is a possible suffix for $\mathbf{p}^{(i)} \cdot \hat{y}^{(i)}_{j..k-1} \cdot y_k$ and \$ is a special symbol that means the end of the hypothesis.

The algorithm works by constructing the output incrementally by appending individual labels. The decision of appending a new label is conditioned to previous labels, but it is independent of future decisions. The idea behind this is that, if the user amends a label at position l, $\hat{y}^{(i)}_l$, all of the following labels in $\hat{\boldsymbol{y}}^{(i)}$ (i.e., $\hat{\boldsymbol{y}}^{(i)}_{l+1..\hat{I}}$) will be discarded in favor of a new suffix, $\hat{\boldsymbol{s}}^{(i+1)}$.

The proof of Prop. 1 provided in Appendix A.1 is inspired in previous work by Oncina [7], who reached a similar algorithm. However, there are two major differences. First, in the SISP protocol proposed in [7], the system predicts one symbol instead of complete suffixes. Second, [7] presented the algorithm for text prediction (cf. autocompletion) where there is no dependency on a structured input $\boldsymbol{x}$. Hence, if the dependence on the $\boldsymbol{x}$ is dropped in Eq. (3), then it is possible to integrate the summation for all suffixes $\boldsymbol{s}'$, producing the optimum algorithm in [7],

$$\hat{y}^{(i)}_k = \operatorname*{argmax}_{y_k} Pr(y_k|\mathbf{p}^{(i)} \cdot \hat{y}^{(i)}_{j..k-1})$$
$$\text{for } k = j \dots \hat{I} \wedge \hat{y}^{(i)}_{\hat{I}+1} \equiv \$ \qquad (4)$$

Thus, the resulting algorithm is *greedy* in the sense that it makes the decision locally, which is in contrast to Eq. (3) where the decision is taken globally. That is, the probability in Eq. (4) models only the next label, whereas in Eq. (3) it models whole suffixes. Unfortunately, for most SP problems, directly modeling the "local" posterior probability in Eq. (4) is still an unresolved problem, especially those with latent variables. Therefore, it is necessary to rely on the global models used in Eq. (3) and perform the sum explicitly over a (potentially) exponential number of suffixes.

3.2. Relation with the MAP Decision Rule

It has been mentioned that the MAP decision rule (Eq. (2)) has been extensively used for SISP. Although Eq. (2) is known to minimize the *zero-one* loss function for hypothesis suffixes, it would be interesting to analyze how it behaves with respect to Eq. (3).

Proposition 2. *The MAP decision rule is equivalent to a maximum approximation to the optimal decision rule for SISP,*

$$\hat{y}^{(i)}_k = \operatorname*{argmax}_{y_k} \max_{\boldsymbol{s}'} Pr(y_k \cdot \boldsymbol{s}'|\boldsymbol{x},\mathbf{p}^{(i)} \cdot \hat{y}^{(i)}_{j..k-1})$$
$$\textit{for } k = j \dots \hat{I} \wedge \hat{y}^{(i)}_{\hat{I}+1} \equiv \$ \qquad (5)$$

The proof of Prop. 2 can be found in Appendix A.2. It follows from a Bayes decomposition of Eq. (2) for each y_k. Prop. 2 has two main implications. Firstly, it provides a formalism for the traditional MAP approach as it can be seen as a maximum approximation to the optimum. That is especially convenient for models where Eq. (3) cannot be computed efficiently (exponential number of suffixes). Secondly, on non-smooth probability distributions where the mass probability is concentrated around the maximum, MAP performs almost as accurately as the optimum algorithm.

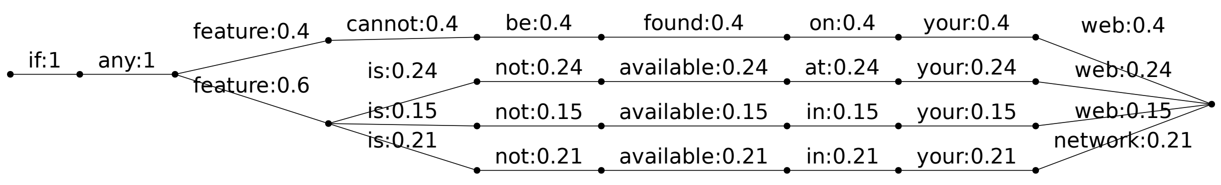


Figure 3: Word graph obtained as the translation of the input sentence in Figure 2. Edges show the hypothesized word and its posterior probability.

4. Decoding Algorithm

The algorithm described in Prop. 1 has the difficulty that the sum over all possible suffixes must be done explicitly. In practice, this may be a major problem, since SP outputs are combinatorial by nature. Hence, to list all possible suffixes can be a hard problem. To deal with this problem, we propose a general practical algorithm for optimal decoding. The whole set of hypotheses (search space) will be represented by a word graph.

A word graph $G(\boldsymbol{x}) = (\Sigma, Q, A, q_i, q_f, f)$ is a probability distribution over the hypothesis space given an input variable $\boldsymbol{x}$. $G(\boldsymbol{x})$ is represented as a directed, acyclic, weighted graph where Σ is a set of symbols and Q is a set of nodes with q_i as the initial node and q_f as the final node. $A : \Sigma \times Q \times Q$ is a set of edges, $e = (y, u, v)$, which hypothesizes a label $y \in \Sigma$ from a start node $u \in Q$ to an end node $v \in Q$ with a score of $f(e, \boldsymbol{x})$. A path $\boldsymbol{w} = (e_1, \cdots, e_J)$ is a sequence of connected edges that represents a complete hypothesis.

Given the input $\boldsymbol{x}$, the posterior probability for a specific edge e can be computed by summing up the posterior probabilities of all hypotheses of the word graph containing e. These posterior probabilities (here we use small p to denote models instead of true probabilities) can be efficiently computed based on the well-known *forward-backward* algorithm [14],

$$p(e|\boldsymbol{x}) = p((y, u, v)|\boldsymbol{x}) = \frac{\Phi(u) f(e, \boldsymbol{x}) \Psi(v)}{\Phi(q_f)} \tag{6}$$

where the forward score $\Phi(u)$ for node u is the sum of all possible paths from the initial node q_i to u. Similarly, the backward score $\Psi(v)$ for node v is the sum of all possible paths from v to the final node q_f. Figure 3 shows a (pruned) word graph obtained as the result of the translation of the input sentence for the example in Figure 2, after the word posterior probabilities have been computed.

Now, we can conveniently introduce the prefix $\boldsymbol{p}^{(i)}$ dependency in Eq. (6):

$$p(e|\boldsymbol{x}, \boldsymbol{p}^{(i)}) = p((y, u, v)|\boldsymbol{x}, \boldsymbol{p}^{(i)}) = \frac{\Phi_{\boldsymbol{p}^{(i)}}(u) f(e, \boldsymbol{x}) \Psi(v)}{Z_{\boldsymbol{p}^{(i)}}} \tag{7}$$

Note that the prefix dependency affects the forward score $\Phi_{\boldsymbol{p}^{(i)}}$. In this case, Eq. (7) is restricted to the sum of all paths from the initial node q_i to u for which the sequence of labels matches the prefix $\boldsymbol{p}^{(i)}$. Also, the normalization factor $Z_{\boldsymbol{p}^{(i)}}$ now only takes into account the mass probability of all the paths that have $\boldsymbol{p}^{(i)}$ as a prefix.

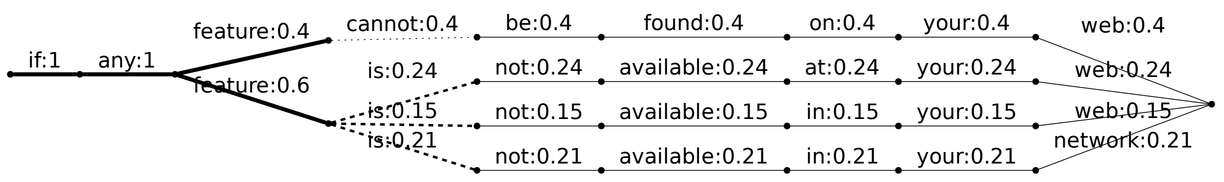
Then, Eq. (3) can be easily computed by marginalizing over all of the edges with the word y that follow $\boldsymbol{p}^{(i)}$:

$$\sum_{\boldsymbol{s}'} Pr(y \cdot \boldsymbol{s}'|\boldsymbol{x}, \boldsymbol{p}^{(i)}) \simeq p(y|\boldsymbol{x}, \boldsymbol{p}^{(i)}) = \sum_{u} \sum_{v} p((y, u, v)|\boldsymbol{x}, \boldsymbol{p}^{(i)}) \tag{8}$$

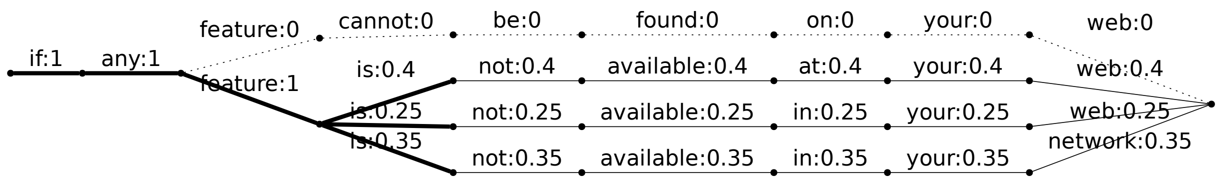
Figure 4 exemplifies how the state of the optimum algorithm changes when predicting the word at position $j = 4$ in iteration $(i = 0)$ for the example in Figure 2. Note that an error is committed by the MAP approach (Figure 4(a)) since it relies on the edge with the highest probability. In contrast, the optimum algorithm selects the set of edges with the same word whose sum is maximum, allowing the correct solution to be chosen.

5. Experimentation

First, we designed a simulated scenario. This allowed us to assess under what conditions the optimal decision rule outperformed the classical approach. For this purpose, we obtained the references from the Wall Street Journal (WSJ)



(a) State of the optimum algorithm at iteration $(i = 0)$ when obtaining $j = 4$. The dashed edges suggest the paths to be chosen by the optimum algorithm, whereas the dotted edge suggests the path to be chosen by the maximum approach.



(b) State of the optimum algorithm at iteration $(i = 0)$ after obtaining $j = 4$. Note that the posterior probabilities have been renormalized to contain only the paths with the prefix 'if any feature is'. Dotted edges display unreachable paths, for which the posterior probability is 0. All the paths that are compatible with the prefix are eligible candidates. Note that, based on the optimum algorithm, the correct output is produced at iteration $(i = 0)$.

Figure 4: Word graphs for the example in Figure 2 for the optimum algorithm. This figure exemplifies how the state of the algorithm changes when predicting the word at position $j = 4$ in iteration $(i = 0)$. Edges show the hypothesized words and the posterior probabilities as in Eq. (8). Bold edges show current compatible prefixes.

database [8]. Then, for each reference we built a graph with 1000 hypotheses generated by introducing uniformly distributed random errors on the reference, with an ε error rate. Next, we assigned a score for each hypothesis assuming that the posterior probability followed an exponential distribution, $\lambda e^{-\lambda n}$, where n is the number of hypotheses and $\lambda \geq 0$ controls the peakedness of the distribution (the bigger the peakier).

In addition, three real world NLP tasks were selected: machine translation, handwritten text recognition, and automatic speech recognition. The problem of these three tasks consists of producing a translation or transcription $\boldsymbol{y}$ given an input signal $\boldsymbol{x}$. Formally, the problem can be solved by obtaining the sentence $\hat{\boldsymbol{y}}$ that maximizes the posterior probability:

$$\hat{\boldsymbol{y}} = \operatorname*{argmax}_{\boldsymbol{y}} Pr(\boldsymbol{y}|\boldsymbol{x}) = \operatorname*{argmax}_{\boldsymbol{y}} Pr(\boldsymbol{x}|\boldsymbol{y}) Pr(\boldsymbol{y}) \tag{9}$$

where $Pr(\boldsymbol{y})$ is typically approximated by a language model, usually n-grams [5]. In contrast, modeling $Pr(\boldsymbol{x}|\boldsymbol{y})$ is task dependent.

5.1. *Machine Translation*

In machine translation (MT), $\boldsymbol{x}$ is a sentence in the source language. $P(\boldsymbol{y}|\boldsymbol{x})$ is usually approximated by phrase-based log-linear models [6]. The experiments were conducted on the Xerox corpus [3] (English and Spanish documents), which is a collection of technical manuals. This corpus features approximately $700k$ running words for training and $90k$ running words for test. Test perplexities are 48 for English and 33 for Spanish. The size of the vocabulary in training is $8k$ and $11k$, respectively, while the number of *out-of-vocabulary* words (OOV) is around 0.6%.

5.2. *Handwritten Text Recognition*

In handwritten text recognition (HTR), $\boldsymbol{x}$ is a feature vector that represents a handwritten sentence. In this case, $Pr(\boldsymbol{x}|\boldsymbol{y})$ represents morphological-lexical knowledge and is approximated by hidden Markov models (HMM) [5]. HTR experiments were conducted on the "Cristo-Salvador" corpus [10], which was kindly provided by the *Biblioteca Valenciana Digital* (BIVALDI)[2]. In the *page* version, the test set is composed of 491 samples corresponding to the last ten lines of each document page ($4.5k$ running words), whereas the training set is composed of the 681 remaining samples ($6.4k$ running words). The experiments were run on a closed vocabulary containing $3.4k$ words.

[2]http://bv2.gva.es

5.3. *Automatic Speech Recognition*

The automatic speech recognition (ASR) problem is modeled in a very similar fashion to the HTR problem. In this case, however, $\boldsymbol{x}$ represents a speech signal. Therefore, $Pr(\boldsymbol{x}|\boldsymbol{y})$ is a HMM model for the phonological-lexical knowledge. The experiments were performed using the WSJ database mentioned in the simulated experiments. Up to 81 hours of training material (WSJ0+WSJ1 partitions) were used to train speaker-independent HMMs. The test was composed of 213 sentences and $3.4k$ running words with a perplexity of 168. The recognition was performed with the open vocabulary setup ($64k$ words) containing 1.6% OOVs.

5.4. *Results*

These corpora were evaluated based on two different measures. First, the PE effort was measured with the *word error rate* (WER). WER can be computed as the edit distance between the hypothesis and the reference normalized by the number of words in the reference. In contrast, SISP systems were evaluated with the *word stroke ratio* (WSR), which is the number of interactions normalized by the number of words in the reference. Note that both the edit distance and the number of interactions could have been computed at the character level as well. In that case, different costs would be attributed to correct shorter or longer predictions, but the influences of the word autocompletion feature that most editors have should also be taken into account. Besides, it may be argued that number of characters corrected seems to be more related to typing effort, whereas the cognitive effort would be more correlated to word units. Hence, word edit distance seems a more intuitive measure. We will denote the traditional MAP approach to SISP as SISP-MAP and the proposed approach as SISP-OPT.

Figure 5 shows the evolution of the SISP-OPT decision rule as λ approaches one. It can be seen that, in this ideal scenario, when the distribution is smooth, the sum of different suffix hypotheses averages to obtain a much better result. However, as λ reaches one the distribution becomes so peaky that SISP-OPT is equal to SISP-MAP from that point on.

The results for real tasks are presented in Table 1. It can be seen that the two SISP systems outperform their PE counterparts. With regard to the comparison between both SISP approaches, SISP-OPT always performs equal to or better than SISP-MAP. As explained in Section 3.2, SISP-MAP should be close or match SISP-OPT performance for non-smooth probability distributions, i.e., the probability mass is concentrated around the maximum. HMMs for HTR are known to have very high ‘peaks’. In fact, as there is only one possible transcription, the true probability distribution would give probability *one* to the reference and *zero* to the rest, resulting in the ‘peakiest’ distribution. This is reflected in the fact that SISP-MAP and SISP-OPT obtain the same exact result. SMT models are also ‘peaky’. However, unlike HTR, there may exist several perfectly correct translation references for a given source sentence. This could explain why SISP-OPT manages to improve an absolute 0.1 for the Xerox English-Spanish corpus. With respect to the ASR results, statistically significant improvements can be also observed, with a *probability of improvement* (poi) of 99% compared to SISP-MAP using the confidence estimation technique proposed in [1]. The reason for this may be that we were able obtain more dense word lattices and, consequently, the summation in Eq. (3) was made over a large set of suffixes.

The results for the real tasks have shown minimal improvements as anticipated in Prop. 2, but they are also supported by [11]. That paper study the relation between the MAP rule and the optimum rule for any integer-valued metric cost function for PE systems. They conclude that the use of optimal decision rules for task-related cost functions has a limited impact. Therefore, although the cost function for SIPS is not a metric by itself, the results indicate that those conclusions could be extrapolated to our case studies.

6. Conclusions

In this paper, we present an optimal decision rule for *sequential interactive structured prediction* (SISP) that generalizes the work on *text prediction* by [7] to SISP problems that depend on a structured input. Our approach extends previous work by allowing full suffix prediction instead of single symbols. This can be considered to be a relevant contribution since they represent the most frequent problems in SP. In addition, the *maximum-a-posteriori* approach was presented as a maximum approximation to the optimal decision rule. Furthermore, a practical and general decoding algorithm was developed over word graphs. Experiments on different NLP tasks have shown that

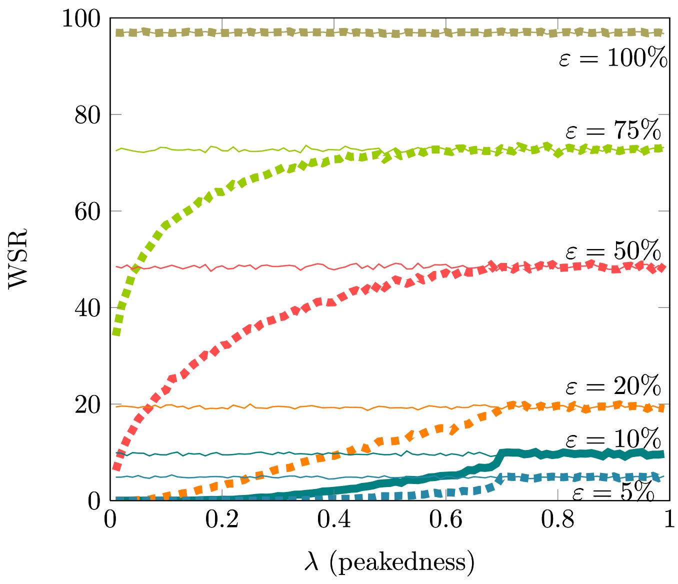


Figure 5: WSR as peakedness increases in simulated experiments for different error rates ε. The thick lines represent the WSR for SISP-OPT, whereas the thin lines represent that of SISP-MAP

Table 1: Results for real tasks. PE represents the post-editing error in a non-interactive scenario. SISP-MAP is the error of the traditional approach to SISP and SISP-OPT is the error of the optimum approach.

	Xerox		CS	WSJ
System	en-es	es-en	es	en
PE (WER%)	24.0	27.0	33.6	15.5
SISP-MAP (WSR%)	23.3	26.3	29.6	13.2
SISP-OPT (WSR%)	23.2	26.3	29.6	13.0[a]

[a]Statistically significant (poi $> 99\%$)

the MAP decision rule performs very similarly to the optimal one for non-smooth probability distributions, as was expected. However, the optimum strategy has still been able to obtain minor improvements.

Further work should delve into the analysis of the optimal decision rule behavior. Directly implementing the optimal decision rule instead of using word graphs would probably lead to better improvements, since the sum is made over a wider range of hypotheses. It would also be interesting to test the results on other NLP tasks. Further research should especially concentrate on finding real tasks with smooth probability distributions so that the behavior of the optimal decision rule under more favorable conditions could be analyzed. Finally, the theoretical properties of the algorithm should also be studied further. Hopefully, that would allow determining under what conditions it is worthwhile to use the optimal decision rule. Thus, if improvements are not expected, then the use of non-optimal algorithms would be completely justified, given that, in practice, MAP algorithms are easier to compute.

Acknowledgements

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 287576 (CasMaCat), and from the Spanish MEC/MICINN under the MIPRCV "Consolider Ingenio 2010" program (CSD2007-00018) and iTrans2 (TIN2009-14511) project. It is also supported by the Generalitat Valenciana under grant ALMPR (Prometeo/2009/01) and GV/2010/067. The authors wish to thank the anonymous reviewers for their criticisms and suggestions.

Appendix A. Proofs

Appendix A.1. Proof of Prop. 1

Proof.

Definition 1. *The SISP protocol has an associated cost function $C(\boldsymbol{y}^{(i)}, j, \boldsymbol{r})$ that computes the cost of sequentially producing a reference $\boldsymbol{r}$ from the label at position j of the hypothesis $\boldsymbol{y}^{(i)}$ at iteration (i). Using the abbreviated notation, $C_j^{(i)}$, the cost can be described as:*

$$C_j^{(i)} = \bar{\delta}_j^{(i)} \left(1 + C_{j+1}^{(i+1)}\right) + \delta_j^{(i)} C_{j+1}^{(i)} \tag{A.1}$$

where $\delta_l^{(i)} = \delta(y_l^{(i)}, r_l)$ is a Kronecker delta *function that is* 1 *if $y_l^{(i)} \equiv r_l$ and* 0 *otherwise, whereas $\bar{\delta}_l^{(i)}$ is the negation of $\delta_l^{(i)}$.*

Following the MCE approach, an optimum algorithm for SIPS is one that minimizes the conditional expected value $(E(\cdot))$ of $C_j^{(i)}$:

$$(\hat{y}_j^{(i)}, \ldots, \hat{y}_{\hat{I}}^{(i)}) \;=\; \underset{(y_j,\ldots,y_I)}{\operatorname{argmin}} \min_I E\left(C_j^{(i)} \middle| \boldsymbol{x}, \boldsymbol{p}^{(i)}\right) \tag{A.2}$$

As the expected value is a linear operator, and after simple transformations, Eq. A.2 becomes:

$$E\left(C_j^{(i)} \middle| \boldsymbol{x}, \boldsymbol{p}^{(i)}\right) = 1 - \sum_{\boldsymbol{s}'} Pr(y_j^{(i)} \cdot \boldsymbol{s}' | \boldsymbol{x}, \boldsymbol{p}^{(i)}) + \sum_{w \neq y_j^{(i)}} E\left(C_{j+1}^{(i+1)} \middle| \boldsymbol{x}, \boldsymbol{p}^{(i)} \cdot w\right) + E\left(C_{j+1}^{(i)} \middle| \boldsymbol{x}, \boldsymbol{p}^{(i)} \cdot y_j^{(i)}\right) \tag{A.3}$$

First, note that the sum of the expected values of $C_{j+1}^{(i)}$ and $C_{j+1}^{(i+1)}$ in Eq. A.3 cover all possible suffixes of $\boldsymbol{p}^{(i)}$. Hence, they are a constant for every possible value of y_j and the minimization can be done independently. Then,

$$\begin{aligned} \hat{y}_j^{(i)} &= \underset{y_j}{\operatorname{argmin}} \left(1 - \sum_{\boldsymbol{s}'} Pr(y_j \cdot \boldsymbol{s}' | \boldsymbol{x}, \boldsymbol{p}^{(i)})\right) \\ &= \underset{y_j}{\operatorname{argmax}} \sum_{\boldsymbol{s}'} Pr(y_j \cdot \boldsymbol{s}' | \boldsymbol{x}, \boldsymbol{p}^{(i)}) \end{aligned} \tag{A.4}$$

Consequently, $\hat{y}_j^{(i)}$ must form part of the optimum hypothesis. The minimization for subsequent elements can be rewritten as

$$(\hat{y}_j^{(i)}, \ldots, \hat{y}_{\hat{I}}^{(i)}) \;=\; \underset{(y_j,\ldots,y_I):y_j \equiv \hat{y}_j^{(i)}}{\operatorname{argmin}} \min_I E\left(C_j^{(i)} \middle| \boldsymbol{x}, \boldsymbol{p}^{(i)}\right) \tag{A.5}$$

Since all but the last term in Eq. A.3 are constant now that $\hat{y}_j^{(i)}$ has been fixed,

$$(\hat{y}_j^{(i)}, \ldots, \hat{y}_{\hat{I}}^{(i)}) \;=\; \underset{(y_j,\ldots,y_I):y_j \equiv \hat{y}_j^{(i)}}{\operatorname{argmin}} \min_I E\left(C_{j+1}^{(i)} \middle| \boldsymbol{x}, \boldsymbol{p}^{(i)} \cdot \hat{y}_j^{(i)}\right) \tag{A.6}$$

Similarly to Eq. A.4, we can obtain $\hat{y}_{j+1}^{(i)}$. Now, by induction it is trivial to prove that, if Eq. A.6 holds for $\hat{y}_{j..k-1}^{(i)}$, it also holds for $\hat{y}_k^{(i)}$, using the same reasoning.That concludes the proof of Prop. 1. □

Appendix A.2. Proof of Prop. 2

Proof. For $k = j$, Eq. (2) and Eq. (5) are obviously equivalent since obtaining y_j from Eq. (2),

$$\hat{y}_j^{(i)} \overset{(2)}{=} \underset{y_j}{\operatorname{argmax}} \max_{y_{j+1..I}} Pr(y_{j..I}|\boldsymbol{x}, \boldsymbol{p}^{(i)}) = \underset{y_j}{\operatorname{argmax}} \max_{y_{j+1..I}} Pr(y_j \cdot y_{j+1..I}|\boldsymbol{x}, \boldsymbol{p}^{(i)}) \quad \text{A.7}$$

for $\boldsymbol{s} = y_{j+1..I}$.

Then, by induction, if both decision rules are equivalent for $\hat{y}_{j..k-1}^{(i)}$, then they are also equivalent for $\hat{y}_k^{(i)}$,

$$\begin{aligned} \hat{y}_k^{(i)} &\overset{(2)}{=} \underset{y_k}{\operatorname{argmax}} \max_{y_{j..k-1}, y_{k+1..I}} Pr(y_{j..I}|\boldsymbol{x}, \boldsymbol{p}^{(i)}) \\ &= \underset{y_k}{\operatorname{argmax}} \max_{y_{j..k-1}, y_{k+1..I}} Pr(y_{j..k-1}|\boldsymbol{x}, \boldsymbol{p}^{(i)}) Pr(y_k \cdot y_{k+1..I}|\boldsymbol{x}, \boldsymbol{p}^{(i)} \cdot y_{j..k-1}) \end{aligned} \quad \text{A.8}$$

Since $\hat{y}_{j..k-1}^{(i)}$ are known to be the optimum values, the first term of the product is constant in Eq. A.8 finally reaching

$$\hat{y}_k^{(i)} \overset{(2)}{=} \underset{y_k}{\operatorname{argmax}} \max_{y_{k+1..I}} Pr(y_k \cdot y_{k+1..I}|\boldsymbol{x}, \boldsymbol{p}^{(i)} \cdot \hat{y}_{j..k-1}) \quad \text{A.9}$$

for $\boldsymbol{s} = y_{k+1..I}$. Therefore, both decision rules are equivalent. $\square$

References

[1] Bisani, M., Ney, H., 2004. Bootstrap estimates for confidence intervals in ASR performance evaluation. In: Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing. Vol. 1. p. 409–412.

[2] Duda, R., Hart, P., Stork, D., Nov. 2001. Pattern Classification, 2nd Edition. Wiley-Interscience.

[3] Esteban, J., Lorenzo, J., Valderrábanos, A., Lapalme, G., 2004. TransType2: an innovative computer-assisted translation system. In: Proceedings of the ACL 2004 on Interactive poster and demonstration sessions. p. 94–97.

[4] Fu, K., 1982. Syntactic pattern recognition and applications. Advanced reference series: Computer science. Prentice-Hall.

[5] Jelinek, F., 1997. Statistical methods for speech recognition. MIT Press.

[6] Och, F., Ney, H., 2002. Discriminative training and maximum entropy models for statistical machine translation. In: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics. p. 295–302.

[7] Oncina, J., 2009. Optimum algorithm to minimize human interactions in sequential computer assisted pattern recognition. Pattern Recognition Letters, 30 (5), 558–563.

[8] Pallett, D., Fiscus, J., Fisher, W., Garofolo, J., Lund, B., Przybocki, M., 1994. 1993 benchmark tests for the ARPA spoken language program. In: Proceedings of the workshop on Human Language Technology. p. 49–74.

[9] Parker, C., Altun, Y., Tadepalli, P. (Eds.), 2009. Machine Learning. Vol. 77. Special issue on structured prediction.

[10] Romero, V., Toselli, A., Rodríguez, L., Vidal, E., 2007. Computer assisted transcription for ancient text images. In: Proceedings of the 4th international conference on Image Analysis and Recognition. p. 1182–1193.

[11] Schlueter, R., Nußbaum-Thom, M., Ney, H., 2012. Does the Cost Function Matter in Bayes Decision Rule? IEEE Transactions on Pattern Analysis and Machine Intelligence, 34 (2), 292–301.

[12] Toselli, A., Vidal, E., Casacuberta, F. (Eds.), 2011. Multimodal Interactive Pattern Recognition and Applications. Springer.

[13] Vidal, E., Rodríguez, L., Casacuberta, F., García-Varea, I., 2007. Interactive pattern recognition. In: Proceedings of the 4th international conference on Machine learning for multimodal interaction. p. 60–71.

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