

Confidence Estimation for NLP Applications

SIMONA GANDRABUR

RALI, Université de Montréal

GEORGE FOSTER

National Research Council, Canada

and

GUY LAPALME

RALI, Université de Montréal

Confidence measures are a practical solution for improving the usefulness of Natural Language Processing applications. Confidence estimation is a generic machine learning approach for deriving confidence measures. We give an overview of the application of confidence estimation in various fields of Natural Language Processing, and present experimental results for speech recognition, spoken language understanding, and statistical machine translation.

Categories and Subject Descriptors: C.4 [**Performance of Systems**]*—Fault tolerance, reliability, availability, and serviceability, modeling techniques*; F.1.1 [**Computation by Abstract Devices**]: Models of Computation*—Self-modifying machines (e.g., neural networks)*; G.3 [**Probability and Statistics**]*—Statistical computing; reliability and robustness*; I.2.6 [**Artificial Intelligence**]: Learning*—Connectionism and neural nets*; I.2.7 [**Artificial Intelligence**]: Natural Language Processing*—Machine translation, speech recognition and synthesis, language parsing and understanding*; J.5 [**Arts and Humanities**]*—Language translation, linguistics*

General Terms: Algorithms, Design, Experimentation, Reliability, Theory

Additional Key Words and Phrases: Confidence measures, confidence estimation, statistical machine learning, computer-aided machine translation, speech recognition, spoken language understanding

1. CONFIDENCE ESTIMATION FOR NATURAL LANGUAGE PROCESSING APPLICATIONS

Despite significant progress in recent years, many NLP technologies are not yet 100% reliable, in the sense that their output is often erroneous. Errors are

Authors' addresses: S. Gandrabur and G. Lapalme, Département d'informatique et de recherche opérationnelle, Université de Montréal, C.P. 6128, Succ. Centre-ville, Montréal, Québec, Canada, H3C 3J7; G. Foster, Language Technologies Research Centre, 283, Alexandre-Taché blvd., P.O. Box 1250, Stn. Hull, Gatineau, Quebec, Canada, J8X 3X7; Corresponding email: lapalme@iro.umontreal.ca.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or direct commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2006 ACM 1550-4875/06/1000-0001 \$5.00

due to the fact that language itself is typically ambiguous and because of technological and technical limitations. Examples are numerous: automatic speech recognition applications (ASR) often return wrong utterances, machine translation (MT) applications frequently produce awkward or incorrect translations, handwriting recognition applications are not flawless, etc.

We describe experiments in *confidence estimation* (CE), a generic machine learning *rescoring* approach for estimating the probability of correctness of the outputs of an arbitrary NLP application. In a general rescoring framework, a baseline application generates a set of alternative output candidates that are then rescored using additional information or models. The new score can then be used as a *confidence measure* for *rejection* or for *reranking*. Reranking consists in reordering the output candidates according to the new score: the top output candidate will be the one that scores highest score as given by the rescoring model. The goal of rejection is different: the new score (also called *confidence score*) is not used to propose an alternative output but to be compared with a *rejection threshold*. Outputs with a confidence score below the threshold are considered insufficiently reliable and are rejected. Because of this fundamental difference between reranking and rejection, rescoring techniques, models and features for reranking and rescoring can differ.

Reranking was successfully applied on a number of different NLP problems, including parsing [Collins and Koo 2005], machine translation [Och and Ney 2002; Shen et al. 2004], named entity extraction [Collins 2002] and speech recognition [Collins et al. 2005; Hasegawa-Johnson et al. 2005].

Confidence measures and confidence estimation methods have been used extensively in ASR for rejection [Gillick et al. 1997; Hazen et al. 2002; Kemp and Schaaf 1997; Maison and Gopinath 2001; Moreno et al. 2001]. Instead of focusing on improving the output of the baseline system, these methods provide tools for effective error detection and error handling. They have proven extremely useful in making ASR applications usable in practice. Confidence measures have been less frequently used in other areas of NLP, arguably because less practical applications are available in these fields: rejection is useful when an end application can determine the cost of a false output versus an incomplete output and set a rejection threshold accordingly.

In Gandrabur and Foster [2003], it was first shown how the same methods used in ASR for CE and rejection could be used in machine translation in the context of TransType, an interactive text prediction application where the cost of a wrong output could be estimated both in practice and by a theoretical user model. In Blatz et al. [2004a], it was noted that CE is a generic approach that could be applied to any technology with underlying accuracy flaws, but results were presented only for machine translation.

Our article gives an overview of CE in general and is a first attempt to present CE experiments in practical applications from different NLP fields, SMT and ASR, under a unified framework.

In Sections 1–4, we set out a general framework for confidence estimation: Section 1 motivates the use of this technique; Section 2 defines confidence measures; Section 3 describes confidence estimation and gives an overview of related work; and Section 4 discusses issues related to evaluation.

In Section 5, we present a case study of the use of CE within an ASR application. Section 6 describes how CE can be used for an interactive MT application. We conclude in Sections 7 and 8 by enumerating the advantages and the limitations of the use of CE in NLP.

1.1 Why are NLP Technologies Difficult?

At a fundamental level, NLP technologies are difficult because the underlying tasks are hard to define precisely. Human judgment about what constitutes a correct answer to a linguistic problem often encompasses many different alternatives, and varies from one person to another. Thus, it can be argued that a notion of confidence is intrinsic even to human language processing: sometimes there are only better or worse responses, and it is expedient to be able to quantify this.

More concretely, there are many reasons why NLP performance is still well below human levels in most areas. Viewed as a learning problem, language is usually characterized by a sparse data condition, one that is exacerbated by wide variations across domains. Statistical models are overly simplistic, both in order to combat over fitting and because the true statistical structure of language is unknown. Practical problems such as noisy inputs and search errors can lead to further degradation of performance.

1.2 Workarounds to Accuracy Flaws

One response to NLP imperfections is to simply ignore them. Many current NLP-based applications adopt this strategy, which calls for careful management of user expectations. For instance, machine translation output is often billed as being useful for *gisting* purposes—sufficient to allow a reader to infer the general scope or meaning of the source text, but not to convey precise details.

Another solution, adopted widely in ASR but less so in other fields of NLP, is the use of *confidence measures*. These are numerical scores assigned to individual outputs that reflect the system's degree of confidence. Confidence measures allow an application to take remedial action when it judges that an error may have occurred. The canonical example in ASR is to prompt a user to repeat a poorly recognized utterance. In a machine-translation post-editing application, confidence measures can be used to identify questionable sentence-level translations for human revision. In information retrieval, they can be used to establish a relevance threshold, beyond which documents will not be shown to the user. In general, confidence measures can make NLP applications more natural and more useful.

In many cases, confidence measures can be derived directly from an NLP system. For instance, the probabilistic scores that statistical systems assign to their outputs can fulfill this role. However, the use of a separate *confidence estimation* component offers more flexibility, because it can easily be recalibrated in response to changes in the operating domain or modifications to the base NLP system. This is the approach to assessing confidence that we emphasize in this article.

2. CONFIDENCE MEASURES

Consider a generic application that for any given input $x \in X$ of a fixed input domain X returns a corresponding output value $y \in Y$ of a fixed output domain Y . Moreover, suppose that this application is based on an imperfect technology, in the sense that the output values y can be correct or false (with respect to some well-defined evaluation procedure). Formally, we can associate a binary tag $C \in \{0, 1\}$, 0 for false and 1 for correct, to each output y , given its corresponding input x . Practically all NLP applications can be cast in this mold, for example:

- Automatic speech recognition (ASR) applications*: x is a speech wave and y is the corresponding sequence of words,
- Machine translation (MT)*: x is a sequence of words in a given source language and y is the equivalent sequence of words in a different target language,
- Automatic language identification (LI)*: x corresponds to text and y is the language the text is written in,
- Information retrieval (IR)*: x is a query for information and y is a document relevant to this query,
- Handwriting recognition*: x is a hand-written text and y is the equivalent typed text.

A *confidence measure* is a function $c : X \times Y \times K \rightarrow I$, where: X and Y are input and output domains as described above; K is an abstract domain of additional knowledge sources used to determine the confidence; and I is the confidence range. Typically, I is an interval $[a, b] \in \mathbb{R}$ such that a lower confidence score is indicative of lower reliability. (Discretizing this interval into specific ranges corresponding to different confidence levels—and ultimately to different actions—is usually left to the end application.)

An important subclass of confidence measures consists of those for which $c(x, y, k)$ is intended to be an estimate of the probability $P(C = 1|x, y, k)$ that y is correct, where $C \in \{0, 1\}$ is a binary *correctness judgment* for y . Such *probabilistic* confidence measures can be used by the end application to choose the action that minimizes risk in a decision-theoretic setting, provided that the costs associated with different scenarios are known. Indeed, if costs depend dynamically on x and y , risk cannot be minimized *without* a probabilistic measure.

Alternative probabilistic confidence measures are posterior probabilities $P(y|x)$, which can often be derived directly from statistical NLP systems. However, these are less general than correctness probabilities $P(C|x, y)$ due to the requirement that $\sum_y P(y|x) = 1$, which forces the maximum probability of correctness to depend on the *number* of correct answers y for a given input x . For problems like machine translation that can have many equally correct answers, correctness probabilities are preferable because they have a uniform interpretation, independent of x .

3. CONFIDENCE ESTIMATION

CE is a generic machine learning approach for computing confidence measures for a given application. It consists in adding a layer on top of the base application that analyzes the outputs in order to estimate their reliability or confidence. In our work, the CE layer is a model for $P(C|x, y, k)$, the correctness probability for output y , given input x and additional information k .

Such a model could, in principle, be used to choose a different output from the one chosen by the base system, but it is not typically used this way (otherwise, the confidence layer could be incorporated to improve the base system). Rather, its role is to predict the probability that best answer(s) output by the base application will be correct. To do this, it may rely on information from k that is irrelevant to the problem of choosing a correct output—for example, the length of the source sentence in machine translation, which is often correlated with the quality of the resulting translation.

3.1 Machine-Learning Approaches

The CE model is discriminatively trained over an annotated corpus that consists of (x, y) -pairs, where x is from the application's input domain and y is the corresponding output produced by the application. To avoid overestimating performance on new text, this corpus should be disjoint from any material originally used to train the application. Each pair in the corpus is annotated with a label $C \in \{0, 1\}$; 1 if y is a correct output, 0 otherwise.

The pair (x, y) is represented as a vector of *confidence features* that captures properties relevant for determining the correctness of the output. Examples of confidence features are:

- the base-model probability estimate (in the case of statistical NLP applications);
- input-specific features (e.g., input sentence length, for a MT task);
- output-specific features (e.g., language-model score computed by an independent language model over an output word sequence);
- features based on properties of the search space (e.g., size of a search graph); and
- features based on output comparisons (e.g., average distance from the current output to alternative hypotheses for the same input).

A detailed classification of types of confidence features is given in Blatz et al. [2004b].

Some of these features—such as the base-model probability—can be used as confidence measures in their own right. With a suitable training corpus and machine learning method, we can expect that a CE layer incorporating many features will perform better than any single feature on its own. Various machine-learning methods can be used to combine features, including stacking [Ting and Witten 1997], boosting [Schapire 2001], SVMs [Joachims 1998], neural nets [Bishop 1995], and Maximum Entropy models [Foster 2000; Och

and Ney 2002]. In this article, we use neural nets to learn combined confidence measures.

3.2 Confidence Estimation with Neural Nets

Our models for confidence estimation are multi-layer perceptrons (MLPs) with a LogSoftMax activation function, trained by gradient descent with a negative-log-likelihood (NLL) criterion. Under these conditions, the outputs of the MLP are guaranteed to be estimates of the conditional correctness probability $p(C = 1|x, y, k)$ [Bishop 1995]. An advantage of MLPs is that the computer resources required for training are modest (unlike SVMs, for example). The ability to train efficiently is important for NLP applications, where the number of training examples may be very large.

We describe experiments with two types of neural nets: MLPs, as described above; and the special case of MLPs that have no hidden layer, which we refer to as single-layer perceptrons (SLPs). This distinction is interesting because MLPs can approximate an arbitrary decision boundary, whereas SLPs are limited to linear decision boundaries. In the following paragraphs, we establish a link, first presented in Gandrabur and Foster [2003], between SLPs and Maximum Entropy models.

For the problem of estimating $p(y|\mathbf{x})$ for a set of classes y over a space of input vectors \mathbf{x} , a single-layer neural net with softmax outputs takes the form:

$$p(y|\mathbf{x}) = \exp(\tilde{\alpha}_y \cdot \mathbf{x} + b) / Z(\mathbf{x})$$

where $\tilde{\alpha}_y$ is a vector of weights for class y , b is a bias term, and $Z(\mathbf{x})$ is a normalization factor, the sum over all classes of the numerator. A maximum entropy model is a generalization of this in which an arbitrary feature function $\mathbf{f}_y(\mathbf{x})$ is used to transform the input space as a function of y :

$$p(y|\mathbf{x}) = \exp(\tilde{\alpha} \cdot \mathbf{f}_y(\mathbf{x})) / Z(\mathbf{x}).$$

Both models are trained using maximum likelihood methods. Given M classes, the maximum entropy model can simulate a SLP by dividing its weight vector into M blocks, each the size of \mathbf{x} , then using $\mathbf{f}_y(\mathbf{x})$ to pick out the y th block:

$$\mathbf{f}_y(\mathbf{x}) = (\mathbf{0}_1, \dots, \mathbf{0}_{y-1}, \mathbf{x}, \mathbf{0}_{y+1}, \dots, \mathbf{0}_M, 1),$$

where each $\mathbf{0}_i$ is a vector of 0's and the final 1 yields a bias term.

The advantage of maximum-entropy models is that their features can depend on the target class. For natural-language applications where target classes correspond to words, this produces an economical and powerful representation. However, for CE, where the output is binary (correct or incorrect), this capacity is less interesting. In fact, there is no a priori reason to use a different set of features for correct outputs or incorrect ones, so the natural form of a maxent model for this problem is identical to a SLP (modulo a bias term). Therefore, our experiments can be seen as comparisons between maxent models and neural nets with a hidden layer.

3.3 Related Work

The most extensive application of CE in NLP is within the field of ASR, where confidence measures are typically used to determine whether or not to ask a user to repeat an utterance, as described below in Section 5. In ASR, approaches based on a separate CE layer tend to predominate, although there is also work on posterior probability confidence measures derived from nbest lists or word lattices [Mangu et al. 2000; Wessel et al. 2001]. Many different machine-learning algorithms have been used for training CE layers, including Adaboost [Carpenter et al. 2001], naive Bayes [Sanchis et al. 2003], Bayesian nets [Moreno et al. 2001], neural networks [Guillevic et al. 2002; Kemp and Schaaf 1997; Weintraub et al. 1997; Zhang and Rudnicky 2001], boosting [Moreno et al. 2001], support vector machines [Ma et al. 2001; Zhang and Rudnicky 2001], linear models [Gillick et al. 1997; Kemp and Schaaf 1997; Moreno et al. 2001], and decision trees [Kemp and Schaaf 1997; Moreno et al. 2001; Neti et al. 1997; Zhang and Rudnicky 2001]. Roughly half of this work concerns CE for probabilistic confidence measures, and half concerns nonprobabilistic measures (i.e., scores derived from binary classifiers). The standard methods for evaluating CE performance, discussed in the next section, have also emerged from work in ASR, for example, Siu and Gish [1999] and Maison and Gopinath [2001].

There is a more recent, but growing, body of work on CE for machine translation and related fields. In Gandrabur and Foster [2003], a neural-net CE layer is used to improve estimates of correctness probabilities for text predictions in a machine-assisted translation tool. Blatz et al. [2004a] report on an extensive study to investigate word- and sentence-level CE on MT output, using Naive Bayes and neural net classifiers; a major conclusion is that noise in automatic MT evaluation metrics makes it difficult to determine correctness at the sentence level. Quirk [2004] suggests the use of human assessments to get around this problem, and finds that training on a small set of human annotated data leads to better performance, when evaluated on similar data, than does training on a much larger automatically annotated corpus. Finally, Ueffing and Ney [2005] improve on the word-level CE techniques of Blatz et al. [2004a] by using a general phrase-based translation model, independent from the base MT system, and by optimizing log-linear parameters in the CE layer to improve confidence error rate directly.

CE has also been used in other areas of NLP. Manmatha and Sever [2002] describe a form of confidence estimation for combining the results of different query engines in information retrieval. Delany et al. [2005] describe an ad hoc ensemble-based approach to combining confidence features in a spam-filtering application. Xu et al. [2002] use a mixture of empirical correctness probabilities for single features (such as answer type) to select a response from either a web-based or a traditional question answering system. Culotta and McCallum [2004] study the problem of confidence estimation in an information extraction system based on linear-chain conditional random fields. They find that posterior probabilities derived using the forward-backward algorithm from the base system's model outperform related ad hoc methods, and give about the same

performance as an external maximum-entropy CE layer that uses the posterior estimates as features. Another important use of CE is in active learning, where correctness probabilities can be used to select informative examples for human annotation Culotta and McCallum [2005]. Finally, CE has applications outside of NLP, for instance in CPU branch prediction for instruction-level parallelism [Grunwald et al. 1998].

4. EVALUATION

In this section, we discuss two aspects of evaluation that are pertinent to CE: evaluation of correctness of the base application outputs and evaluation of the accuracy and usefulness of confidence measures.

4.1 Evaluation of Correctness

As described above, the approach to CE that we promote in this article requires examples of system input and output labelled with binary correctness judgments. These are used for both training and evaluation purposes.

Producing such data requires extensive human effort and is subject to typical problems such as noisy input and inter-annotator disagreement. Nevertheless, it is relatively straightforward for problems such as speech recognition and text classification for which there is at least nominally one right answer. In such cases, standard data sets labeled with the correct answer can be established, and the correctness status for a given system's output over these data sets can be determined automatically.

The situation is more complicated for problems like machine translation and automatic summarization for which the answers are multiple and less clearly defined. Here the best approach would be to have humans evaluate all system outputs, but this is often too expensive, as it needs to be done separately for different systems and even for different versions of the same system. An alternative is to establish standard data sets containing a *set* of correct answers for each input, from which the status of a system output can be calculated using some distance measure such as the well-known BLEU metric in machine translation [Papineni et al. 2001]. Binary correctness judgments can then be derived by thresholding this distance measure.

4.2 Evaluation of Confidence Measures

Since the motivation for using confidence measures is to improve an application, the ultimate gauge of their success is the gain in end-to-end performance. However, it is also useful to have metrics that allow for more direct evaluation of confidence measures. We use two such metrics:

- Receiver Operating Characteristic (ROC) curves*, and related quantitative metrics for assessing discriminability [Zou 2004]; and
- Normalized Cross Entropy (NCE)* for measuring the relative decrease in uncertainty about the correctness of the application outputs produced by the confidence measure [NIST 2001].

ROC curves apply to any confidence score (probabilistic or not), and are valuable for comparing discriminability and determining optimal rejection thresholds for specific application settings. NCE applies only to probabilistic measures, and tests the accuracy of their probability estimates; it is possible for such measures to have good discriminability and poor probability estimates, but not the converse. In the following sections, we briefly describe each metric.

4.2.1 ROC Curves. Any confidence measure $c(x, y, k)$ can be turned into a binary classifier by choosing a threshold θ and judging the system output y to be correct whenever $c(x, y, k) \geq \theta$. Over a given dataset consisting of pairs (x, y) labelled for correctness as defined in Section 3.1, the performance of the classifier can be measured using standard statistics like accuracy. The statistics used for ROC are *correct acceptance* (CA) and *correct rejection* (CR): the proportion of pairs scoring above θ that are in fact correct, and the proportion scoring below θ that are incorrect.

In general, different values of θ will be optimal for different applications. An ROC curve summarizes the global performance of $c(x, y, k)$ by plotting $CR(\theta)$ versus $CA(\theta)$ as θ varies. An advantage of using CA and CR for this purpose is they range from 0 to 1 and 1 to 0, respectively, as θ ranges from the smallest value of $c(x, y, k)$ in the dataset to the largest. This means that ROC curves always connect the points (1,0) and (0,1) and thus are easy to compare visually (in contrast to other plots such as recall versus precision). Better curves are those that run closer to the top right corner, and the best possible connects (1,0), (1,1), and (0,1) with straight lines. This corresponds to perfect discriminability, with all correct points (x, y) assigned higher scores than incorrect points. Curves “below” the diagonal are equivalent to those above, with the sign of $c(x, y, k)$ reversed.

A quantitative metric related to ROC curves is IROC, the integral below the curve, which ranges from 0.5 for random performance to 1.0 for perfect performance. This metric is similar to the Wilcoxon rank sum statistic [Blatz et al. 2004b]. Alternatively, one can choose a fixed operating point for CA (for instance, 0.80), and compare the corresponding CR values given by difference confidence measures.

4.2.2 Normalized Cross Entropy. NIST introduced the Normalized Cross Entropy (NCE) measure as the cross entropy (mutual information in this context) between the correctness of the system’s output y and its probabilistic confidence score, normalized by a baseline cross entropy:

$$\left[H_{base} + \sum_{\text{correct } y} \log_2 P(C = 1|x, y) + \sum_{\text{incorrect } y} \log_2 P(C = 0|x, y) \right] / H_{base},$$

where $H_{base} = n \log_2 p_c + (N - n) \log_2 (1 - p_c)$, p_c is the prior probability of correctness, and n is the number of correct outputs out of N total outputs.

The idea of NCE is to allow for comparison of confidence measures across different test sets by correcting for the effect of prior probabilities, which tend to boost raw cross-entropy scores when they are high. The term H_{base} is proportional to the maximum cross entropy obtainable by assigning a fixed probability

of correctness to all examples when the prior is p_c . NCE scores normally range from 0 when performance is at this baseline to 1 for perfect probability estimates, although negative scores are also possible.

5. CONFIDENCE ESTIMATION FOR AUTOMATIC SPEECH RECOGNITION

This section describes the use of CE within an ASR application, first presented in Guillevis et al. [2002], where a human user interacts by speaking with a system, in our case a directory assistant application.

The user says an *utterance*, either a phrase or a short command, and the ASR application interprets the utterance into a sequence of words, called *recognition hypothesis*. Then, a Spoken Language Understanding (SLU) module extracts meaning from the recognition hypothesis by interpreting it into a sequence of *concepts* or *semantic slots*.

The example below illustrates the interaction between a human user and an ASR directory application. A user wanting to reach a certain “McPhearson” on the telephone, utters the desired name when prompted by the application. The ASR engine returns three alternative recognition hypotheses as a N-best set:

- 1 McPhearson
- 2 Paul McPhearson
- 3 Mike Larson

A robust parser transforms each recognition hypothesis into one or more *semantic hypotheses* or *semantic interpretations*, that is, a sequence of concepts:

- 1 (userID=103, LastName=McPhearson)
- 2 (userID=103, FullName=Paul McPhearson)
- 2 (userID=103, LastName=McPhearson)
- 3 (userID=217, FullName=Mike Larson)
- 3 (userID=217, LastName=Larson)
- 3 (userID=315, LastName=Larson)

A concept is a key-value pair; in our example, the keys are userID, LastName and FullName. For instance, the second recognition hypothesis “Paul McPhearson” has two different semantic interpretations: both corresponding to the same destination (userID=103), one corresponding to a full parse (FullName=Paul McPhearson) and the other only a partial parse that leaves out the first name “Paul” (LastName=McPhearson).

A dialog manager then decides that the confidence level of the destination concept userID=103 is sufficiently high to transfer the call this destination.

This application relies on concept-level confidence scores for designing its dialog strategies. For instance, if the confidence level of a destination is too low, the application might decide to prompt the user for a confirmation, such as “Would you like to talk to Paul McPhearson?”. Alternatively, if all confidence levels of semantic interpretations are too low, the application might decide to reject the entire utterance and prompt for a repetition, such as: “Sorry, I did not understand. Could you please repeat your request?”.

Typical confidence measures for ASR applications first determine the reliability of the ASR results at the word level and then at the utterance level [Wessel et al. 2001; Stolcke et al. 1997; Sanchis et al. 2003; Carpenter et al.

2001; Hazen et al. 2000; Moreno et al. 2001; Zhang and Rudnicky 2001]. However, in speech understanding systems, recognition confidence must be reflected at the semantic level in order to be able to apply various confirmation strategies directly at the concept level. In the previous example, the application is not concerned about how well the word “Paul” was recognized when deciding to transfer the call to Paul McPhearson. It only needs to take into account how confident the destination userID=103 is.

Concept-level confidence scores can be obtained by computing word and utterance confidence scores and feeding them to the natural language processing unit in order to influence its parsing strategy [Hazen et al. 2000]. This approach is mostly justified when semantic interpretation tends to be a time-consuming processing task: the time to complete a robust parsing over a large N-best list (or equivalent word-lattice) is reduced by only focusing on areas of the word-lattice where recognition confidence is high.

Alternatively, a confidence score can be computed directly at the concept level [San-Secundo et al. 2001]. This method has the advantage of being more complete: it enables taking parsing-specific and semantic consensus information into account when computing the concept-level confidence score.

In the following experiments, we first compute a confidence score at the word level, based mainly on acoustic features computed over N-best lists (or the equivalent word-lattices). Second, we parse the utterances and use the word-level confidence score (among other features) to compute a concept-level confidence score.

Confidence scores are computed using the CE approach: we train MLPs to estimate the probability of correctness of recognition results on two levels of granularity: word-level and concept-level. The training is supervised, based on an annotated corpus consisting of various utterances and their corresponding recognition and semantic hypotheses, annotated as correct or wrong at word-level, respectively concept-level.

We use the same corpus for both CE tasks, consisting of 40,000 utterances. We split our corpus into three sets: 49/64 of the data for training, 7/64 for validation and the remaining 8/64 for testing.

5.1 Word-Level CE for ASR

For each utterance in the corpus, we generate ASR results as sets of N-best recognition hypotheses with $N = 5$.

A per-word timing information is available, that indicates the precise start and end *frame* of the speech signal (the minimal unit the signal is divided into) that each word corresponds to. This timing information allows us to recover the word-lattice structure from the N-best lists.

Each word in a recognition hypothesis constitutes a training/testing example for the word-level CE task: a number *confidence features* and a *correct/wrong tag* are computed for each word.

A word is tagged as correct if it can be found in roughly the same position in the utterance’s reference transcription and incorrect otherwise. This step is performed by aligning the hypothesis and the reference word sequences using a

Table I. Correct Rejection (CR) Rate for an Acoustic Word Score Classifier Using a MLP Network with Two Nodes in the Hidden Layer with a Correct Acceptation Rate of 95%

Score	NCE	CR
Frame-weighted acoustic score	0.25	34%
CE word score	0.32	47%

dynamic programming algorithm with equal cost for insertions, deletions, and substitutions. As a result, each hypothesized word is tagged as either: *insertion*, *substitution*, or *ok*, with the *ok* words tagged as *correct*, while the others are tagged as *incorrect*. Note that the comparison between two words is based on pronunciation, not orthography.

We use four different word-level confidence features:

Frame-Weighted Raw Score. This corresponds to the log-likelihood that the recognized word is correct given the speech signal that it was aligned to, as estimated by the probabilistic acoustic models used within the ASR application. This score is computed by averaging the *acoustic score* (the log-likelihood estimated by the acoustic models used in the ASR system that the recognized word-segment is correct given the speech signal) over all frames in a word.

Frame-Weighted Acoustic Score. This is the frame-weighted raw score normalized by an on-line *garbage model* [Caminero et al. 1996]. It is the base word score originally used by the underlying ASR application;

State-Weighted Acoustic Score. This score is similar with the frame weighted acoustic score. The difference is that the score is averaged not over frames but over Hidden Markov Model (HMM) states corresponding to the word;

Word Length. The number of frames of each word.

Given the four word-level features mentioned above, plus the correct tag, we train a MLP network with two nodes in the hidden layer and a LogSoftMax activation function, using gradient descent with a negative-log-likelihood (NLL) criterion. In choosing the MLP architecture (the number of hidden units and layers) we are making a trade-off between: classification accuracy and robustness: more complex architectures (with more hidden units and/or and additional layer) seemed to over fit the data, even when early training stopping was performed, and were more sensitive to small changes in the training corpus. The MLP was trained to estimate the probability of a word being correct, according to its confidence features. Prior to being fed to the network, the features are scaled by removing the mean and projecting the feature vectors on the eigenvectors computed on the pooled-within-class covariance matrix. The pooled matrix is the one used in Fisher's Linear Discriminant Analysis.

As shown in Table I, at a correct acceptance (CA) rate of 95%, we are able to boost our correct rejection (CR) rate from 34% to 47%. This represents a 38% relative increase in our ability to reject incorrect words. We also measured the relative performance of our new word score using the Normalized Cross Entropy (NCE) introduced by NIST. We are able to boost the entropy from 0.25 to 0.32.

This means that the new word-level confidence score is more discriminative than the original per-word acoustic score provided by the ASR engine. This improvement leads to better discriminability at the concept level, as will be described in the following section.

5.2 Concept-Level CE for SLU

Using the same experimental set-up as in the preceding section on word-level CE for ASR, we applied the CE approach for SLU, namely at the concept (semantic slot) level.

The start and end times of a semantic slot are respectively defined as the start time of the first word and the end time of the last word that make up the slot. This timing information allows us to obtain a concept-level lattice corresponding to each recognition hypothesis N-best list produced by the ASR engine.

Prior to describing the concept-level confidence features, we define the notion of *normalized hypothesis a posteriori probabilities* used in the definition of our confidence features.

5.2.1 Normalized Hypothesis a Posteriori Probabilities. The recognition hypotheses (sometimes also referred to as *sentences*) within an N-best set are ordered according to their *sentence score*, that is the log-likelihood of the joint probability of the acoustic scores and the language model probabilities of the word sequence: the higher the sentence score, the higher the estimated probability of correctness of the hypothesis and the higher its rank within the N-best set.

The sentence score is very appropriate for ranking hypotheses, but does not give an indicative absolute confidence measure because it is greatly affected by the sentence length: it is a point probability computed by multiplying the individual word acoustic and language scores. Therefore, longer sentences do not necessarily mean lower probability of correctness. We define the normalized hypothesis score as the posterior probability to obtain a measure that is more robust to sentence length.

Given a set of N-best recognition hypotheses H_i with their corresponding sentence scores S_i , we compute the *normalized hypothesis a posteriori probabilities* \hat{p}_i according to the following formula:

$$p_i = \exp[k \Delta_i], \quad \text{and} \quad \hat{p}_i = p_i / \sum_{j=0}^N p_j.$$

where the constant k has been set experimentally to 0.075 in our applications, and $\Delta_i = S_i - S_0$:

In Table II, we illustrate the computation of a posteriori hypothesis probabilities \hat{p}_i on an example of three recognition hypotheses and their corresponding sentence scores S_i .

In the following sections, we will first describe our concept-level confidence features, then the labeling method for these concepts and finally our experimental results.

Table II. Computation of a Posteriori Hypothesis Probabilities on Three Recognition Hypotheses and the Corresponding Sentence Scores

Rank	Sentence	S_i	Δ_i	$k \Delta_i$	p_i	\hat{p}_i
1	McPhearson	-5610.114	0.0	0.0	1.0	0.9033
2	Paul McPhearson	-5647.917	-37.803	-2.835	0.0587	0.0530
3	Mike Larson	-5650.505	-40.391	-3.029	0.0483	0.0436

5.2.2 Confidence Features. For each semantic slot, we consider six confidence features:

- word confidence score *average*;
- word confidence score *standard deviation*;
- semantic consensus*;
- time consistency*;
- fraction of unparsed words* per hypothesis;
- fraction of unparsed frames* per hypothesis.

The last two confidence features are computed for an entire semantic hypothesis. Semantic consensus and time consistency are similar to *slot a posteriori probabilities*. Mangu et al. [2000] and Wessel et al. [2001] use a similar scheme to compute word posteriors and consensus-based confidence features.

Word Confidence Score Average and Standard Deviation are computed over all words in a slot.

Semantic Consensus is a measure of how well a semantic slot is represented in the N-best ASR hypotheses. For each semantic slot value, we sum the normalized hypothesis a posteriori probabilities of all hypotheses containing that slot. The measure does not take into account the position of the slot within the hypothesis.

Time Consistency is related to the semantic consensus described above. For a given slot, we add the normalized a posteriori probabilities of all hypotheses where the slot is present and where the start/end times match. The start and end frames for each hypothesized word are given by the ASR and then translated into a normalized time scale. The start and end times of a semantic slot are then respectively defined as the start time of the first word and the end time of the last word that make up the slot.

Fraction of Unparsed Words and Frames indicates how many of the hypothesized words are left unused when generating a given slot value. The classifier will learn that a full parse is often correlated with a correct interpretation, and a partial parse with a false interpretation. The measure of unparsed frames is similar to the measure of unparsed words, with the difference that the words are weighted by their length in number of frames.

5.2.3 Tag—Classifier. We tag each semantic slot as either “correct” or “incorrect”. A semantic slot is “correct” if it appears in the list of slots associated with the reference, regardless of its position.

We feed the six measures mentioned above into an MLP with five hidden neurons. The features are pre-processed before being fed to the network by

Table III. CA/CR Rates Obtained with Semantic Slot Classifiers Trained on Various Subsets of Confidence Features^a

Features	NCE	CR		
		CA = 90%	CA = 95%	CA = 99%
base-line: AC-av	0.32	59%	36%	6%
CE-av, CE-sd	0.35	64%	46%	15%
SC	0.48	81%	66%	21%
AC-av, CE-av/sd, SC, TC	0.55	86%	70%	21%
all	0.56	87%	71%	27%

^aThe first column gives the features used in each case with the following abbreviations: AC-av = word acoustic score average, CE-av = word confidence score average, CE-sd = word confidence score standard deviation, SC = semantic consensus, TC = time consistency FUF = fraction of unparsed words, FUF = fraction of unparsed frames. The second column gives the normalized cross-entropy (NCE). The last three columns give the correct rejection rates at a given correct acceptance rate (CA).

Table IV. Impact of Normalized a Posteriori Hypothesis Weights (hypAPostProb) on CA/CR Rates. SC = semantic consensus, TC = time consistency

Features	NCE	CR
		CA = 90%
SC, TC without hypAPostProb	0.39	68%
SC, TC with hypAPostProb	0.48	81%

removing the mean and projecting the feature vector on the eigenvectors of the pooled-within-class covariance matrix.

5.2.4 Results. We train different MLPs using various subsets of features. We measure the performance of the resulting semantic slot classifiers by analyzing the correct rejection (CR) ability at various correct acceptance (CA) rates given in Table III.

We see that, by feeding our six measures into a MLP, we enhance the performance of a baseline that uses only the average of the original frame-weighted acoustic word scores. The entropy jumps from 0.32 to 0.56 and the CR rate goes from 36% to 71% for a CA rate of 95%.

In the computation of the *semantic consensus* and *time consistency* measures, the occurrence rates of slots in the N-best set are weighted by the normalized hypothesis a posteriori probabilities. In order to measure the impact of these weights, we build classifiers based on these features computed with and without the weights, assuming equal probabilities for all hypotheses. Results are presented in Table IV and show an important impact of the weights on the discriminability of the classifiers.

We have shown how a generic machine learning approach to CE can improve the usefulness of an ASR application. The CE layer produces a combined confidence score that estimates the probability of correctness of an output. This score could be used to discriminate effectively between correct and wrong outputs. We now show how the same approach can be applied to a Machine Translation task.

6. CONFIDENCE ESTIMATION FOR MACHINE TRANSLATION

Confidence measures have been introduced in Statistical Machine Translation (SMT) [Gandrabur and Foster 2003] in the context of TransType, an interactive

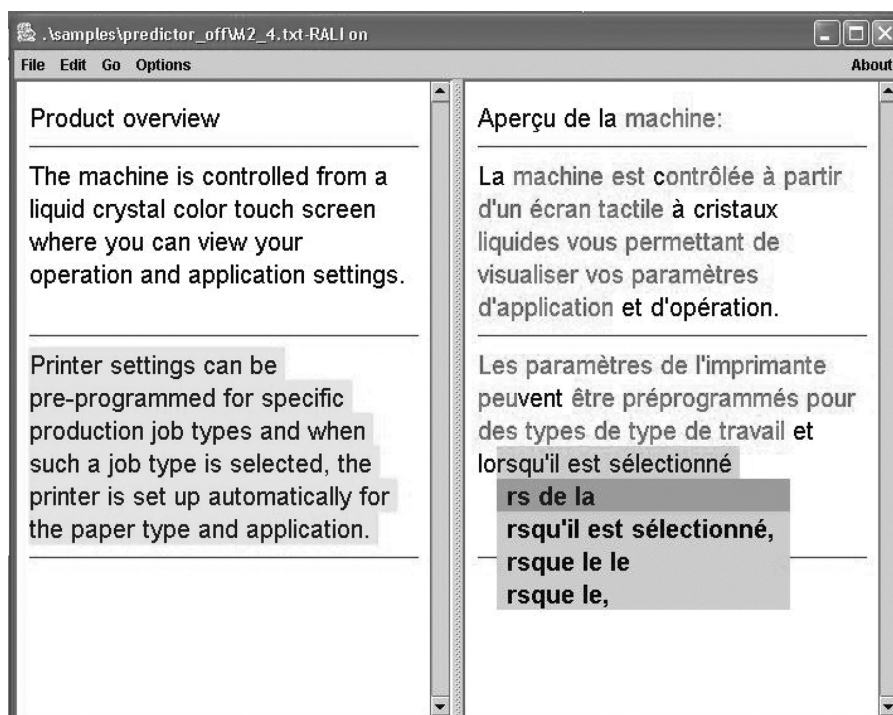


Fig. 1. Screen shot of a TransType prototype. Source text is on the left half of the screen, whereas target text is typed on the right. The current proposal appears in the target text at the cursor position.

text prediction application, then by [Ueffing et al. 2003]. CE for SMT was the subject of a JHU CLSP summer workshop in 2003 [Blatz et al. 2004a, 2004b].

The application we are concerned with is an interactive text prediction tool for translators, called *TransType*¹ [Foster et al. 1997; Langlais et al. 2004].

The system observes a translator as he or she types a text and periodically proposes extensions to it, which the translator may accept as is, modify, or ignore simply by continuing to type. With each new keystroke the translator enters, TransType recalculates its predictions and attempts to propose a new completion to the current target text segment. Figure 1 shows a TransType screen during a translating session.

There are several ways in which such a system can be expected to help a translator:

- when the translator has a particular text in mind, the system’s proposals can speed up the process of typing it
- when the translator is searching for an adequate formulation, the proposals can serve as suggestions
- when the translator makes a mistake, the system can provide a warning,

¹More information about the project and an interactive demonstration is available at <http://rali.iro.umontreal.ca/Traduction/TransType.en.html>.

either implicitly by failing to complete the mistake or explicitly by attaching an annotation for later review,

The novelty of this project lies in the mode of interaction between the user and the machine translation (MT) technology on which the system is based. In previous attempts at interactive MT, the user has had to help the system analyze the source text in order to improve the quality of the resulting machine translation. While this is useful in some situations (e.g., when the user has little knowledge of the target language), it has not been widely accepted by translators, in part because it requires capabilities (in formal linguistics) that are not necessarily those of a translator. Using the target text as the medium of interaction results in a tool that is more natural and useful for translators.

This idea poses a number of technical challenges. First, unlike standard MT, the system must take into account not only the source text but that part of the target text which the user has already keyed in. This section will mainly explore how the tool can limit its proposals to those in which it has reasonable confidence but we have also explored how it should adapt its behavior to the current context, based on what it has learned from the translator [Nepveu et al. 2004].

If the system's suggestions are right, then reading them and accepting them should be faster than performing the translation manually. However, if prediction suggestions are wrong, the user loses valuable time reading through wrong predictions before disregarding them. Therefore, the system must ideally find an optimal balance between presenting the user with predictions or "acknowledging its ignorance" and not propose unreliable predictions. Further complications arise from the fact that longer predictions are typically less reliable. On the other hand, a long correct prediction saves more typing time if accepted, just as a long wrong prediction is longer to read and therefore wastes more user-time. For this reason, we adopted a flexible prediction length. Suggestions may range in length from 0 characters to the end of the target sentence; it is up to the system to decide how much text to predict in a given context, balancing the greater potential benefit of longer predictions against a greater likelihood of being wrong, and a higher cost to the user (in terms of distraction and editing) if they are wrong or only partially right.

An avant-garde tool like this one has to be tested and its impact evaluated in authentic working conditions, with human translators attempting to use it on realistic translation tasks. We have conducted user evaluations and collected for assessing the strengths and weaknesses both of the system and the evaluation process itself. Part of this data was used for comparing different combinations of parameters in the following experimentations.

In the project's final evaluation rounds [Macklovitch 2006], most of the translators participating in the trials registered impressive productivity gains, increasing their productivity from 30 to 55% thanks to the completions provided by the system. The trials also served to highlight certain modifications that would have to be implemented if the TransType prototype is to be made into a translator's every day tool: e.g. the ability of the system to learn from the users' corrections, in order to modify its underlying translation models.

6.1 CE for TransType

Our CE approach for TransType consists in training neural nets to estimate the conditional probability of correctness

$$p(C = 1 | \hat{\mathbf{w}}_m, \mathbf{h}, \mathbf{s}, \{\mathbf{w}_m^1, \dots, \mathbf{w}_m^n\})$$

where $\hat{\mathbf{w}}_m = \mathbf{w}_m^1$ is the most probable prediction of length m from a N-best set of alternative predictions according to the base model.

In our experiments, the prediction length m varies between 1 and 4 and n is at most 5. As the N-best predictions $\{\mathbf{w}_m^1, \dots, \mathbf{w}_m^n\}$ are themselves a function of the context, we will note the conditional probability of correctness by $p(C = 1 | \hat{\mathbf{w}}_m, \mathbf{h}, \mathbf{s})$.

The data used in our experiments originates from a Hansard English-French bi-text corpus. In order to generate the train and test sets, we use 1.3 million (900 000 for training and 400 000 for testing purposes) translation predictions for each fixed prediction length of one, two, three and four words, summing to a total of 5.2 million prediction examples characterized by their confidence feature vector.

Each translation prediction of the test and training corpus is automatically tagged as correct or incorrect. A translation prediction \mathbf{w}_m is tagged as correct if an identical word sequence is found in the reference translation, properly aligned. This is a very crude and severe tagging method. It could be relaxed by using multiple reference translations and more sophisticated ways of determining whether two translations are equivalent, rather than requiring identity.

For each original translation model, we experimented with two different CE model architectures: MLPs with one hidden layer containing 20 hidden units and one with 0 hidden units, also referred to as single layer perceptron (SLP).

Moreover, for each native model, CE model architecture pair, we train five separate CE models: one for each fixed prediction length of one, two, three or four words, and an additional model for variable prediction lengths of up to four words.

Training and testing of the neural nets was done using the open-source machine-learning library Torch [Collobert et al. 2002].

6.1.1 Confidence Features. The features can be divided into three families (Table V):

- F1 capture the intrinsic difficulty of the source sentence \mathbf{s} ;
- F2 reflect how hard \mathbf{s} is to translate in general;
- F3 reflect how hard \mathbf{s} is for the current model to translate.

For the first two families, we used two sets of values: static ones that depend on \mathbf{s} and dynamic ones that depend on only those words in \mathbf{s} that are still untranslated, as determined by an IBM2 word alignment between \mathbf{s} and \mathbf{h} . The features are listed in Table V. The average number of translations per source word were computed according to an independent IBM1 model (features 9,14). Features 13 and 18 are the density of linked source words within the aligned region of the source sentence.

Table V. Confidence Features^a

	Family	Feature	IROC	
			Maxent1	Bayes
0	F1	static trigram perplexity	0.3879	0.4156
1		static min trigram word probability	0.5567	0.5623
2		static av. word frequency	0.5680	0.5593
3		static av. word length	0.4734	0.4774
4		dynamic trigram perplexity	0.3868	0.4178
5		dynamic min trigram word probability	0.5673	0.5965
6		dynamic av. word frequency	0.5879	0.5263
7		dynamic av. word length	0.4659	0.4500
8		nb. of words	0.4873	0.4312
9	F2	static av. nb. of transl. per source word	0.4644	0.4841
10		static av. source word entropy	0.4460	0.4536
11		static nb. of token to be translated	0.4873	0.4312
12		static nb. o unknown words	0.5011	0.5008
13		static source words density	0.5725	0.5352
14		dynamic av. nb. of transl. per source word	0.5148	0.5511
15		dynamic av. source word entropy	0.4875	0.4774
16		dynamic nb. of source token to be translated	0.5041	0.3538
17		dynamic nb. of unknown words	0.4948	0.4929
18		dynamic source words density	0.3987	0.4403
19		length of current target prefix	0.4503	0.5635
20	F3	prediction length	0.2776	0.2181
21		av. nb. of pruned search hypotheses	0.5608	0.5293
22		search lattice size	0.2931	0.2772
23		active vocabulary size	0.4630	0.4292
24		number of N-best hypothesis	0.3791	0.3230
25		index of current hypothesis	0.5000	0.5000
26		posterior hypothesis probability	0.8070	0.7799
27		base model probability	0.8595	0.8019

^aDescription and discriminability for predictions of $m = 1, \dots, 4$ words. The second column describes the families of features : F1 features capture the intrinsic difficulty of the source sentence; F2 features reflect how hard \mathbf{s} is to translate; F3 features reflect how hard \mathbf{s} is for the current model to translate. The last two columns are the IROC values obtained with each feature.

6.2 Experimental Results

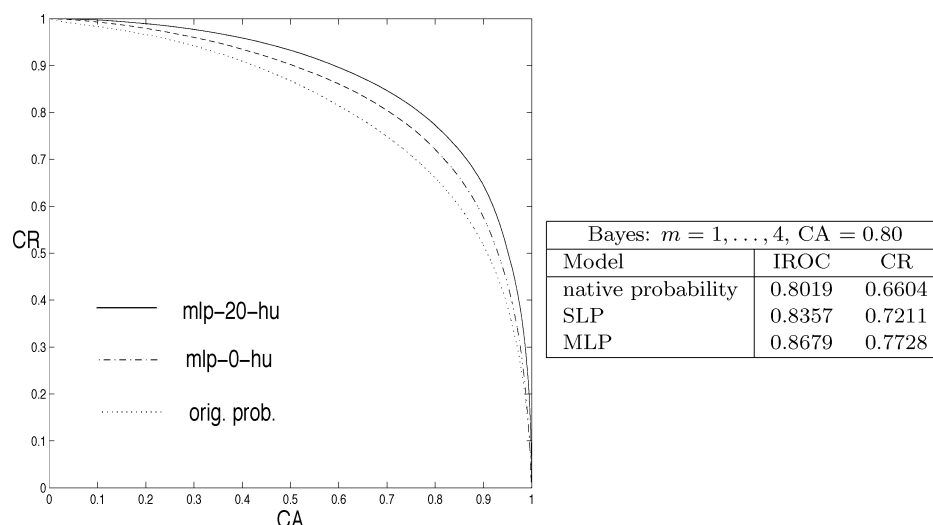
We performed experiments to test the discriminability of scores (Sections 6.2.1 – 6.2.3). To compare the discrimination capacity of scoring functions, we use ROC curves and the *integral* of the ROC curve, or IROC introduced in Section 4.2.1. We also compare scoring functions by fixing an operational point at $CA = 0.80$ and observing the corresponding CR values.

To establish usability improvements in terms of timing speed due to the use of the CE layer we use a user-model simulation [Foster et al. 2002] (Section 6.2.4).

Finally, we will test the usefulness of the CE layer in a different setting: for combining translation prediction outputs from separate independent translation models through a simple maximum confidence score voting scheme (Section 6.2.5).

6.2.1 Discriminability: CE versus and Native SMT Probabilities. In our setting, the tokens are the $\hat{\mathbf{w}}_m$ translation predictions and the base-model score function is the conditional probability $p(C = 1 | \hat{\mathbf{w}}_m, \mathbf{h}, \mathbf{s})$.

Table VI. Bayes model, number of words $m = 1, \dots, 4$: ROC Curve and comparison of discrimination capacity between the Bayes prediction model probability and the CE of the corresponding SLP and MLP on predictions of up to four words



The first question we wish to address is whether we can improve the correct/false discrimination capacity by using the probability of correctness estimated by the CE model instead of the native probabilities.

For each SMT model, we compare the ROC plots, IROC and CA/CR values obtained by the native probability and the estimated probability of correctness output by the corresponding SLPs (also noted as mlp-0-hu) and the 20 hidden units MLPs on the one-to-four words prediction task.

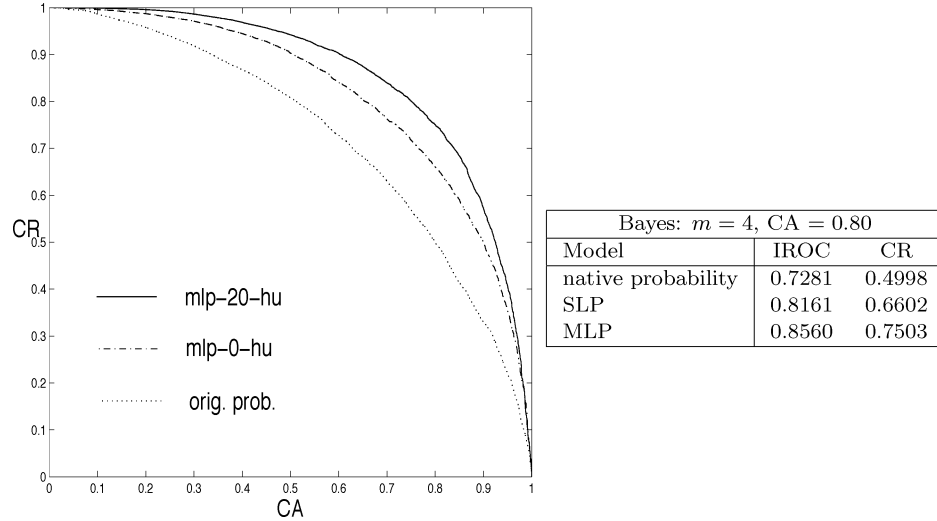
Results obtained for various length predictions of up to four words using the Bayes models are summarized in Table VI. At a fixed CA of 0.80, we obtain the CR increases from 0.6604 for the native probability to 0.7211 for the SLP and 0.7728 for the MLP. The overall gain can be measured by the relative improvements in IROC obtained by the SLP and MLP models over the native probability, that are respectively 17.06% and 33.31%.

The improvements obtained in the fixed-length four-words-prediction tasks with the Bayes model, shown in Table VII, are even more important: the relative improvements on IROC are 32.36% for the SLP and 50.07% for MLP.

However, the results obtained in the Maxent models are not so good, arguably because the Maxent models are already discriminantly trained. The SLP CR actually drops slightly, indicating that linear separation was too simplistic and that additional features only added noise to the already discriminant Maxent score. The MLP CR increases slightly to a 4.80% relative improvement in the CR rate for the Maxent1 model and only 3.9% for the Maxent2 model (Table VIII). The results obtained with the two Maxent models being similar, we only give the ROC curve for the Maxent2 model.

It is interesting to note that the native model prediction accuracy did not affect the discrimination capacity of the corresponding probability of correctness

Table VII. Bayes model, number of words $m = 4$: ROC curve and comparison of discrimination capacity between the Bayes prediction model probability and the CE of the corresponding SLP and MLP on fixed-length predictions of four words



of the CE models, see Table IX. Even though the Bayes model accuracy and IROC is much lower than the Maxent model's, the CE IROC values are almost identical.

6.2.2 Relevance of the Confidence Features. We investigate the relevance of the different confidence features for correct/incorrect discrimination and how they vary when we change the underlying SMT models.

We computed the IROC values obtained by the individual confidence features in the one-to-four words prediction task using the Maxent1 and the Bayes models and present them in Table V.

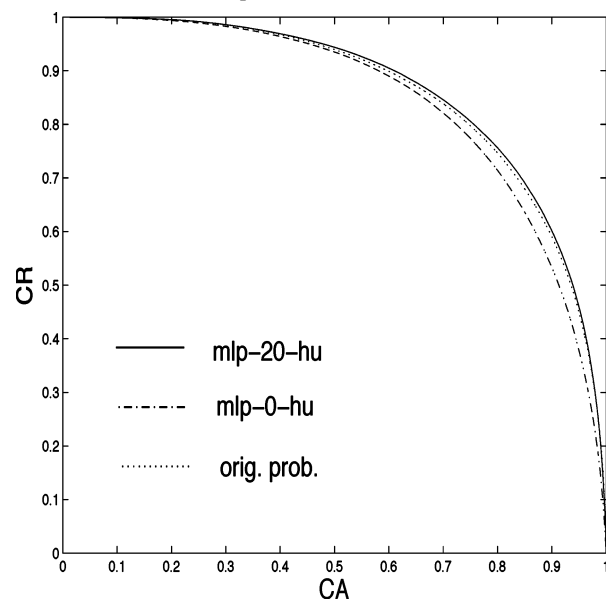
The group of features that performs best over both models are the model- and search-dependent features (21, ..., 27), followed by the features which capture the intrinsic difficulty of the source sentence and the target-prefix (0, ..., 8, 20), and the least relevant ones are the remaining features related to the translation difficulty.

The most relevant feature is the native probability (27), followed by the posterior hypothesis probability (26) and the prediction length (20).

However, there are several interesting differences between the two models, mostly related to the length of the translations (16, 19, 20): the Bayes models seem to be much more sensitive to long translations than the Maxent models.

6.2.3 Dealing with Predictions of Various Lengths. We try different approaches for dealing with various length predictions: we train four separate MLPs for fixed length predictions of one, respectively, two, three, or four words. Then, we train a unique MLP over predictions with various lengths, ranging from one to four, and compared the results in Table X

Table VIII. Comparison of Discrimination Capacity between the Maxent1 and Maxent2 Prediction Models Probability and the CE of the Corresponding SLP and MLP on Predictions of up to Four Words



Model	Maxent1		Maxent2	
	IROC	CR	IROC	CR
native probability	0.8581	0.7467	0.8595	0.7479
SLP	0.8401	0.7142	0.8352	0.6973
MLP	0.8636	0.7561	0.8638	0.7599

Table IX. Discrimination vs. Prediction Accuracy:
Number of Words $m = 1, \dots, 4$

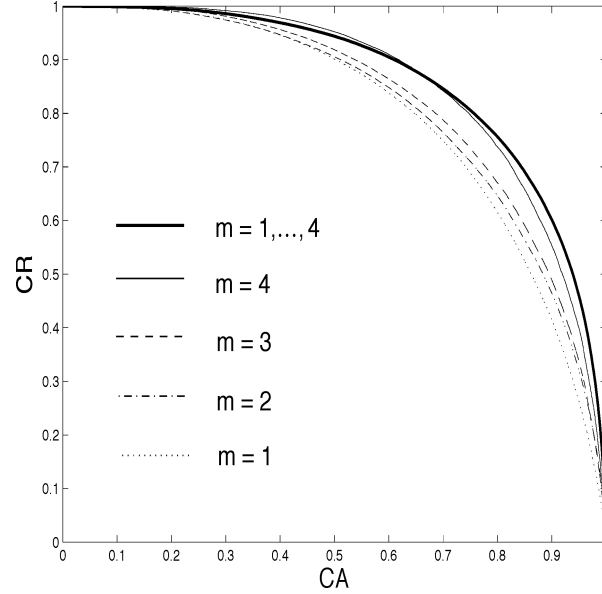
SMT Model	IROC Native		
	%C=1	Probability	IROC MLP
Bayes	8.77	0.8019	0.8679
Maxent1	21.65	0.8581	0.8636
Maxent2	22.23	0.8595	0.8638

The best discriminability was obtained by the MLP trained on predictions of various lengths and in all cases the CE-score discriminability is at least as good as the base score (the native probability estimated by the base models).

6.2.4 User-Model Evaluations. Evaluations with the user simulation are performed only with the Maxent1 and Maxent2 models. The results on a single Hansard file containing 1272 sentence pairs are shown in the Table XI. All are generated using the mixed length (1–4 word predictions) neural nets.

According to our experiments, the best performing systems are the ones using the MLPs with 20 hidden units. These systems outperformed the baseline system that used the base model probabilities during the simulation for

Table X. Model = Maxent2, nb. of Words as $m = 1, m = 2, m = 3, m = 4, m = 1, \dots, 4$: ROC Curve and Impact of Prediction Length on Discrimination Capacity and Accuracy for the Maxent2 Prediction Model



Discrimination vs. prediction length: Maxent2			
Prediction length m	%C = 1	IROC native probability	IROC CE MLP
$m = 1$	44.78	0.7926	0.7986
$m = 2$	23.30	0.8074	0.8121
$m = 3$	13.12	0.8261	0.8245
$m = 4$	7.74	0.8517	0.8567
$m = 1, \dots, 4$	22.23	0.8595	0.8638

typing-speed benefit computations. Interestingly, the simpler CE-models with 0 hidden units performed worse than the baseline.

6.2.5 Model Combination. We now describe how various model combination schemes affect prediction accuracy. We use the Bayes and the Maxent2 prediction models: we try to exploit the fact that these two models, being fundamentally different, tend to be complementary in some of their responses. Adding the Maxent1 model to the combination would not have yield additional improvements, as Maxent1 and Maxent2 are too similar. We chose the latter because it has a slightly better prediction accuracy than the first. Our CE models are the corresponding MLPs because they clearly outperform the SLPs. The results presented in Table XII are reported on the various length prediction task for up to four words.

The combination schemes are the following: we run the two prediction models in parallel and choose one of the proposed prediction hypotheses according to

Table XI. User Simulation Results with Various CE-Layer Configurations^a

Model	CE Layer	Typing Time SSaved
Maxent1	base	11.4%
	all/00	10.9%
	all/20	12.4%
	sel/00	9.6%
	sel/20	12.8%
Maxent2	base	13.3%
	all/00	12.4%
	all/20	13.4%
	sel/00	12.7%
	sel/20	13.6%

^aThe *base* entries are for the base model; *all* means the full feature sets, *sel* means selected ones; and the numbers after the slashes describe the hidden units. The results column gives estimated percentage of typing time saved according to the user model.

Table XII. Prediction Accuracy of the Bayes and Maxent2 Model Compared with Combined Model Accuracy

Model Combination Prediction Accuracy	
Prediction Model Combination	Accuracy
Bayes alone	8.77
Maxent alone	22.23
Max native probability vote combination	17.49
Max CE vote combination	23.86
Optimal combination	27.79

the following voting criteria:

- Maximum CE vote: choose the prediction with the highest CE;
- Maximum native probability vote: choose the prediction with the highest native probability.

As a baseline comparison, we use the accuracy of the individual native prediction models. Then we compute the maximum gain we can expect with an optimal model combination strategy, obtained by running an “oracle” that always picks the right answer.

The results are positive: the maximum CE voting scheme obtains a 29.31% of the maximum possible accuracy gain over the best of the two individual models (Maxent2). Moreover, if we choose the maximum native probability vote, the overall accuracy actually drops. These results are a strong motivation for our post-prediction confidence estimation approach: by training an additional CE layer using the same confidence features and train data for different underlying prediction models we obtain more uniform estimates of the probability of correctness. Contrary to the native probabilities, the CE probabilities are compatible among the various underlying models and can be used for model combinations.

The combination scheme is very basic and relies on maximum score voting. Other combination schemes (such as stacking Section 3.1) might be more

adequate for the task but we wanted to show another possible use of the already defined confidence scores.

We conclude that CE can be successfully used for Transtype: productivity gains can be observed during simulations when confidence score is used for deciding which translation predictions to present to the user, instead of using the base model probabilities. Moreover, the CE-based combination scheme, although very simplistic, allow us to combine the results of two independent translation systems exceeding the performance of each individual system.

7. DISCUSSION: CE ADVANTAGES AND LIMITATIONS

As described in Section 1, confidence measures can either be derived directly from the base system or calculated by a separate CE layer. In this section, we compare these two approaches.

Confidence measures that come directly from the base system have the obvious advantage of simplicity, particularly for probabilistic systems where scoring functions can be used without modification. Apart from the work needed to create a CE layer itself, effort is also required to produce the necessary training corpus of examples of system output labelled with their correctness status, as discussed in Section 4. For some tasks, like machine translation, this corpus is substantially more difficult to produce than the one used to train the base system, which may be already available (e.g., a parallel corpus).

A key advantage of using a CE layer is that is a separate module, distinct from the base system. This has several implications:

- Because it specializes in solving the confidence problem, it often gives better results than confidence measures derived directly from the base system, as we have seen in this article.
- It can incorporate sources of knowledge that may be difficult or inconvenient to add to the base system. These include features like sentence length for MT that are good predictors of confidence but poor predictors of the correct output. Because it is applied in a post-processing step, a CE layer can also model properties of the search algorithm that are inaccessible to the base system.
- The CE layer is smaller and lighter than the base system, and hence easier to adapt to new domains and new users (modulo the availability of correctness data for these contexts).
- It is possible to re-use CE software for different base systems.

The other main advantage to using a CE layer is that it can compute estimates of correctness probabilities $P(C = 1|x, y, k)$. The best alternatives from the base system are posterior probabilities $p(y|x)$, but these are often difficult to extract, and they can be poor estimates. As described in Section 2, they are also less useful than correctness probabilities due to the normalization constraint. Correctness probabilities are useful in many contexts:

- In a decision-theoretic setting, probabilities are required in order to calculate the lowest-cost output for a given input.

- When a collection of heterogeneous models is available for some problem, output probabilities provide a principled way of combining them.
- When multiplying conditional probabilities to compute joint distributions, the accuracy of the result is crucially dependent on the stability of the conditional estimates across different contexts. This is important for applications like speech recognition and machine translation that perform searches over a large space of output sentences, represented as sequences of words.

Thus, for many applications, the benefits from using a separate CE layer will justify the extra effort involved.

8. CONCLUSION

We presented CE as a generic machine learning approach for computing confidence measures for improving an NLP application. CE can be used if relevant confidence features are found, if reliable automatic evaluation methods of the base NLP technology exist and if large annotated corpora are available for training and testing the CE models.

Applications benefit from a light CE-layer when the base application is fixed. If potential base-model accuracy improvements are poor, the additional data is more useful in practice for improving discriminability and removing wrong outputs. We showed that the probabilities of correctness estimated by the CE layer exceed the original probabilities in discrimination capacity.

CE can be used to combine results from different applications. In our MT application, a model combination based on a maximum CE voting scheme yielded a 29.31% accuracy improvement in maximum accuracy gain. A similar voting scheme using the native probabilities decreased the accuracy of the model combination.

Prediction accuracy was not an important factor for determining the discrimination capacity of the confidence estimate. However, the original probability estimation was the most relevant confidence feature in most of our experiments.

We experimented with multi-layer perceptrons (MLPs) with 0 (similar to Maxent models) and up to 20 hidden units. Results varied with the amount of data available and the task, but overall, the use of a hidden units improved the performance.

We intend to continue experimenting further with CE on other SMT systems for other NLP tasks, such as language identification or parsing.

ACKNOWLEDGMENTS

The authors wish to thank Didier Guillevic and Yves Normandin for their contribution in providing the experimental results regarding the ASR application we presented. We wish to thank the members of the RALI laboratory for their support and advice, in particular Elliott Macklovitch and Philippe Langlais.

REFERENCES

BISHOP, C. M. 1995. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, England.

ACM Transactions on Speech and Language Processing, Vol. 3, No. 3, October 2006.

- BLATZ, J., FITZGERALD, E., FOSTER, G., GANDRABUR, S., GOUTTE, C., KULESZA, A., SANCHIS, A., AND UEFFING, N. 2004a. Confidence estimation for machine translation. In *Proceedings of COLING 2004* (Geneva). 315–321.
- BLATZ, J., FITZGERALD, E., FOSTER, G., GANDRABUR, S., GOUTTE, C., KULESZA, A., SANCHIS, A., AND UEFFING, N. 2004b. Confidence estimation for machine translation. Tech. Rep., Final report JHU / CLSP 2003 Summer Workshop, Baltimore, MD.
- CAMINERO, J., DE LA TORRE, D., VILLARUBIA, L., MARTIN, C., AND HERNANDEZ, L. 1996. On-line garbage modeling with discriminant analysis for utterance verification. In *Proceedings of the 4th ICSLP 1996*. Vol. 4. Philadelphia, PA, 2115–2118.
- CARPENTER, P., JIN, C., WILSON, D., ZHANG, R., BOHUS, D., AND RUDNICKY, A. 2001. Is this conversation on track? In *Eurospeech 2001* (Aalborg, Denmark). 2121–2124.
- COLLINS, M. 2002. Ranked algorithms for named entity extraction: Boosting and the voted perceptron. In *Proceedings of the 40th Annual Meeting of the ACL*. Philadelphia, PA, 489–496.
- COLLINS, M. AND KOO, T. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics* 1 (Mar.), 25–70.
- COLLINS, M., ROARK, B., AND SARACLA, M. 2005. Discriminative syntactic language modeling for speech recognition. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*. Association for Computational Linguistics, Ann Arbor, MI. 507–514.
- COLLOBERT, R., BENGIO, S., AND MARIÉTHOZ, J. 2002. Torch: A modular machine learning software library. Tech. Rep. IDIAP-RR 02-46, IDIAP.
- CULOTTA, A. AND MCCALLUM, A. 2004. Confidence estimation for information extraction. In *Proceedings of HLT-NAACL*.
- CULOTTA, A. AND MCCALLUM, A. 2005. Reducing labeling effort for structured prediction tasks. In *Proceedings of AAAI*.
- DELANY, S. J., CUNNINGHAM, P., AND DOYLE, D. 2005. Generating estimates of classification confidence for a case-based spam filter. Tech. Rep. TCD-CS-2005-20, Trinity College Dublin Computer Science Department.
- FOSTER, G. 2000. A Maximum Entropy / Minimum Divergence translation model. In *Proceedings of the 38th Annual Meeting of the ACL* (Hong Kong). 37–42.
- FOSTER, G., ISABELLE, P., AND PLAMONDON, P. 1997. Target-text mediated interactive machine translation. *Mach. Trans.* 12, 1-2, 175–194. TransType.
- FOSTER, G., LANGLAIS, P., AND LAPALME, G. 2002. User-friendly text prediction for translators. In *Conference on Empirical Methods in Natural Language Processing (EMNLP-02)*, (Philadelphia, PA). 148–155.
- GANDRABUR, S. AND FOSTER, G. 2003. Confidence estimation for translation prediction. In *Proceedings of CoNLL-2003*, W. Daelemans and M. Osborne, Eds. Edmonton, B.C., Canada, 95–102.
- GILLICK, L., ITO, Y., AND YOUNG, J. 1997. A probabilistic approach to confidence measure estimation and evaluation. In *ICASSP 1997*. 879–882.
- GRUNWALD, D., KLAUSER, A., MANNE, S., AND PLESZKUN, A. R. 1998. Confidence estimation for speculation control. In *Proceedings of the 25th International Symposium on Computer Architecture*. 122–131.
- GUILLEVIC, D., GANDRABUR, S., AND NORMANDIN, Y. 2002. Robust semantic confidence scoring. In *Proceedings of the 7th International Conference on Spoken Language Processing (ICSLP) 2002* (Denver, CO).
- HASEGAWA-JOHNSON, M., BAKER, J., BORYS, S., CHEN, K., COOGAN, E., GREENBERG, S., JUNEJA, A., KIRCHHOFF, K., LIVESCU, K., MOHAN, S., MULLER, J., SONMEZ, K., AND WANG, T. 2005. Landmark-based speech recognition: Report of the 2004 Johns Hopkins Summer Workshop. In *ICASSP*.
- HAZEN, T., BURIANEK, T., POLIFRONI, J., AND SENEFF, S. 2000. Recognition confidence scoring for use in speech understanding systems. In *Proc. Automatic Speech Recognition Workshop* (Paris, France). 213–220.
- HAZEN, T., SENEFF, S., AND POLIFRONI, J. 2002. Recognition confidence scoring and its use in speech understanding systems. *Compu. Speech Lang.*, 49–67.
- JOACHIMS, T. 1998. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of ECML-98, 10th European Conference on Machine Learning*, C. Nédellec and C. Rouveirol, Eds. Lecture Notes in Computer Sciences, vol. 1398. Springer Verlag, Heidelberg, DE, Chemnitz, DE, 137–142.

- KEMP, T. AND SCHAAF, T. 1997. Estimating confidence using word lattices. In *Eurospeech*. 827–830.
- LANGLAIS, P., LAPALME, G., AND LORANGER, M. 2004. Transtype: Development-evaluation cycles to boost translator's productivity. *Machine Translation (Special Issue on Embedded Machine Translation Systems 17)*, 77–98.
- MA, C., RANDOLPH, M., AND DRISH, J. 2001. A support vector machines-based rejection technique for speech recognition. In *ICASSP 2001*.
- MACKLOVITCH, E. 2006. TransType2: The last word. In *Proceedings of LREC 2006*. Genoa, Italy.
- MAISON, B. AND GOPINATH, R. 2001. Robust confidence annotation and rejection for continuous speech recognition. In *ICASSP 2001*.
- MANGU, L., BRILL, E., AND STOLCKE, A. 2000. Finding consensus in speech recognition: Word error minimization and other applications of confusion networks. *Compu. Speech Lang.* 14, 4, 373–400.
- MANMATHA, R. AND SEVER, H. 2002. A formal approach to score normalization for meta-search. In *Proceedings of HLT 2002, 2nd International Conference on Human Language Technology Research*, M. Marcus, Ed. Morgan Kaufmann, San Francisco, CA. 98–103.
- MORENO, P., LOGAN, B., AND RAJ, B. 2001. A boosting approach for confidence scoring. In *Proceedings of Eurospeech 2001* (Aalborg, Denmark). 2109–2112.
- NEPVEU, L., LAPALME, G., LANGLAIS, P., AND FOSTER, G. 2004. Adaptive language and translation models for interactive machine translation. In *EMLNP'04* (Barcelona, Spain). 190–197.
- NETI, C., ROUKOS, S., AND EIDE, E. 1997. Word-based confidence measures as a guide for stack search in speech recognition. In *ICASSP 1997*. 883–886.
- NIST. 2001. Automatic evaluation of machine translation quality using n -gram co-occurrence statistics. www.nist.gov/speech/tests/mt/doc/ngram-study.pdf.
- OCH, F. J. AND NEY, H. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting of the ACL* (Philadelphia, PA). 295–302.
- PAPINENI, K., ROUKOS, S., WARD, T., AND ZHU, W. 2001. Bleu: A method for automatic evaluation of machine translation. Technical Report RC22176 (W0109-022), IBM Research Division, Thomas J. Watson Research Center (2001).
- QUIRK, C. B. 2004. Training a sentence-level machine translation confidence measure. In *Proceedings of LREC* (Lisbon, Portugal). 825–828.
- SAN-SECUNDO, R., PELLOM, B., HACIOGLU, K., WARD, W., AND PARDO, J. 2001. Confidence measures for spoken dialogue systems. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-2001)* (Salt Lake City, UT).
- SANCHIS, A., JUAN, A., AND VIDAL, E. 2003. Improving utterance verification using a smoothed naive Bayes model. In *ICASSP 2003*, (Hong Kong). 29–35.
- SCHAPIRE, R. 2001. The boosting approach to machine learning: An overview. In *Proceedings of the MSRI Workshop on Nonlinear Estimation and Classification* (Berkeley, CA, Mar.).
- SHEN, L., SARKAR, A., AND OCH, F. J. 2004. Discriminative reranking for machine translation. In *HLT-NAACL*. 177–184.
- SIU, M. AND GISH, H. 1999. Evaluation of word confidence for speech recognition systems. *Comput. Speech Lang.* 13, 4, 299–318.
- STOLCKE, A., KOENIG, Y., AND WEINTRAUB, M. 1997. Explicit word error minimization in N-best list rescoring. In *Proceedings of the 5th European Conference on Speech Communication and Technology*. Vol. 1. (Rhodes, Greece). 163–166.
- TING, K. M. AND WITTEN, I. H. 1997. Stacking bagged and dagged models. In *Proceedings of the 14th International Conference on Machine Learning* (Nashville, TN). Morgan Kaufmann, San Matco, CA. 367–375.
- UEFFING, N., MACHERY, K., AND NEY, H. 2003. Confidence measures for statistical machine translation. In *Proceedings of the 6th Conference of the Association for Machine Translation in the Americas* (New-Orleans, LA). Springer-Verlag, New York, 394–401.
- UEFFING, N. AND NEY, H. 2005. Word-level confidence estimation for machine translation using phrase-based translation models. In *Proceedings of HLT/EMNLP*. Vancouver, BC, Canada.
- WEINTRAUB, M., BEAUFAYS, F., RIVLIN, Z., KONIG, Y., AND STOLCKE, A. 1997. Neural-network based measures of confidence for word recognition. In *ICASSP 1997*. 887–890.

- WESSEL, F., SCHLUTER, R., MACHERY, K., AND NEY, H. 2001. Confidence measures for large vocabulary continuous speech recognition. *IEEE Trans. Speech Audio Process.* 9, 3, 288–298.
- XU, J., LICUANAN, A., MAY, J., MILLER, S., AND WEISCHEDEL, R. 2002. Trec2002qa at bbn: Answer selection and confidence estimation. In *Proceedings of the 10th Text REtrieval Conference (TREC 2002)*.
- ZHANG, R. AND RUDNICKY, A. 2001. Word level confidence annotation using combinations of features. In *Eurospeech*. Aalborg, Denmark, 2105–2108.
- ZOU, K. H. 2004. Receiver operating characteristic (ROC) literature research. <http://splweb.bwh.harvard.edu:8000/pages/ppl/zou/roc.html>.

Received October 2004; revised March 2006; accepted May 2006 by Kevin Knight