Available online at www.sciencedirect.com
ScienceDirect
Computer Speech and Language 22 (2008) 148–170
COMPUTER SPEECH AND LANGUAGE
www.elsevier.com/locate/csl

Practical use of non-local features for statistical spoken language understanding ☆

Minwoo Jeong *, Gary Geunbae Lee

Department of Computer Science and Engineering, Pohang University of Science & Technology (POSTECH), San 31, Hyoja-Dong, Pohang 790-784, Republic of Korea

Received 1 August 2006; received in revised form 4 July 2007; accepted 9 July 2007
Available online 20 July 2007

Abstract

Spoken language understanding (SLU) addresses the problem of mapping natural language speech to frame structure encoding of its meaning. The statistical sequential labeling method has been successfully applied to SLU tasks; however, most sequential labeling approaches lack long-distance dependency information handling method. In this paper, we exploit non-local features as an estimate of long-distance dependencies to improve performance of the statistical SLU problem. A method we propose is to use trigger pairs automatically extracted by a feature induction algorithm. We describe a light practical version of the feature inducer for which a simple modification is efficient and successful. We evaluate our method on three SLU tasks and show an improvement of performance over the baseline local model.

Keywords: Spoken language understanding; Non-local features; Long-distance dependency; Feature induction

1. Introduction

Understanding spoken language has been extensively studied in the last decade by both language processing and speech communities. The goal of spoken language understanding (SLU) is to extract meanings from natural language speech. The SLU covers many spoken language problems such as utterance classification, speech summarization, and information extraction from speech. In particular, the SLU has received much recent interests for a part of spoken dialog system to infer intentions of the speaker for providing natural human–computer dialog interface (Price, 1990; Peckham, 1991; Walker et al., 2002). For spoken dialog system, SLU aims to fill the domain-specific frame slots from speech recognized utterance. Unlike written language, spoken language is much noisier and more ungrammatical; therefore, the techniques for language understanding that are invented on text documents are not robust in spoken language dialog. Hence, a major challenge to SLU is finding a robust solution, which is independent of text-based information such as a full syntactic structure.

☆ The preliminary and initial version of this paper was previously published in COLING/ACL 2006 conference poster session (Jeong and Lee, 2006).

* Corresponding author. Tel.: +82 54 279 5581; fax: +82 54 279 2299.
E-mail addresses: stardust@postech.ac.kr (M. Jeong), gblee@postech.ac.kr (G.G. Lee).

0885-2308/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.csl.2007.07.001

The paradigm proposed so far for SLU shifts from knowledge-based to statistical approaches. Statistical SLU can automatically learn from training examples with corresponding semantics. While rule-based approach often achieves good performance in commercial systems, data-driven approach is more robust and feasible because it is more portable and less expensive to build semantic labeled data for a new domain. Moreover, the statistical framework is growing into a more powerful tool for SLU by using modern machine learning methods: the system can reduce the labeling efforts by active/semi-supervised learning (Tur et al., 2005), and can incorporate prior knowledge into statistical model (Schapire et al., 2002; Wang et al., 2005a).

In this paper, we focus on SLU for spoken dialog system, especially the slot-filling problem. We solve statistical SLU as a sequential labeling problem where we assume that input sequence **x** and output sequence **y** are significantly correlated by a left-to-right sequential structure (Dietterich, 2002; Sutton and McCallum, 2006). For most sequential labeling problems in natural language processing (NLP), a decision is made based on local information. However, the processing which relies on the Markovian assumption cannot efficiently represent higher-order dependencies. This long-distance dependency problem has been considered at length in computational linguistics, which is the key limitation in making better sequential models in various natural language tasks. Thus, we need a new method to import *non-local* information into sequential models.

There are two types of methods for using non-local information for sequential labeling task. One is to use a more complex structure to allow higher-order dependencies and another is to add features to encode the non-locality. To represent higher-order dependencies, modeling the recursive structure of language using stochastic context free grammars has been focused (Fine et al., 1998; He and Young, 2005). In named entity recognition (NER), an additional consistent edge of a linear-chain conditional random field (CRF) explicitly models the higher-order dependencies between distant occurrences of similar words (Sutton and McCallum, 2004; Finkel et al., 2005). However, this approach requires additional time complexity in inference and learning, and the latter is only suitable for representing constraints by enforcing label consistency. In general, the hierarchical structure and higher-order Markov model can naturally capture the non-local dependency; however, it is computationally costly or sometime intractable. We hope to develop a practical method to identify ambiguous labels with general dependency without additional time cost in inference and learning.

Another approach to modeling non-locality is to use observational features which can capture non-local information. Traditionally, many systems prefer to use a syntactic parser or domain-specific knowledge for this purpose. In a language understanding task, the head word dependencies or parse tree path are successfully applied to learn and predict semantic roles, especially those with ambiguous labels (Gildea and Jurafsky, 2002). Although the power of syntactic structure is impressive, using the parser-based feature fails to integrate correct global information due to the low accuracy of a modern statistical parser (Collins, 1999; Charniak and Johnson, 2005). Furthermore, the inaccurate result of parsing is more serious in an SLU task. In contrast to written language, spoken language loses much information including grammar, structure or morphology and even contains some errors in automatically recognized speech. Another preferable solution is to use domain-specific dictionary or heuristic rules. In ATIS domain, well engineered features are used to improve performance (Wang et al., 2005b), but using domain knowledge reduces a merit of data-driven approach; domain portability.

To solve the above problems, we present an efficient and reliable method to exploit non-local information – the trigger features. In this paper, we incorporate trigger pairs into a sequential model, a linear-chain CRF. Then we describe an efficient algorithm to extract the trigger features from the training data itself. The framework for inducing trigger features is based on the Kullback–Leibler divergence criterion which measures the improvement of log-likelihood on the current parameters by adding a new feature (Della Pietra et al., 1997). To reduce the cost of feature selection, we suggest a new modified version of the inducing algorithm, which is quite efficient. We evaluate our method on SLU tasks, and demonstrate the improvements on both transcripts and speech recognition outputs. On a real-world problem, our modified version of a feature selection algorithm is very efficient for both performance and time complexity.

The organization of this paper is as follows. In Section 2, we describe a statistical SLU in terms of the sequential labeling problem and debate long-distance dependency on statistical SLU. In Section 3, we describe an approach to exploit non-local trigger features and an efficient trigger induction algorithm. In Section 4, we present our experimental results. First, we compare our method with other non-local models, and show that trigger features improve performance on both text and spoken inputs. Then, we perform an extensive compar-

ison of trigger selection methods, demonstrating that our new modified algorithm improves performance and reduces time cost. Next, we perform experiments for two non-hierarchical SLU tasks, in which our method is useful for improving performance of a practical spoken dialog system. We discuss previous related works and compare them with our method in Section 5. Finally, we conclude in Section 6.

2. Statistical spoken language understanding

2.1. Spoken language understanding as a sequential labeling problem

The goal of SLU is to extract semantic meanings from recognized utterances and to fill the correct values into a semantic frame structure. A semantic frame is a well-formed and machine-readable structure of extracted information consisting of slot/value pairs. An example of such a reference frame for air travel data (CU-Communicator corpus) is as follows.

<s> i wanna go from denver to new york on november eighteenth </s>
FROMLOC.CITY_NAME = denver
TOLOC.CITY_NAME = new york
MONTH_NAME = november
DAY_NUMBER = eighteenth

In this example, the slot labels are two-level hierarchical; such as FROMLOC and CITY_NAME. This hierarchy differentiates the semantic frame extraction problem from the conventional NER problem in information extraction fields. In fact, this hierarchy causes the non-local dependency problem in SLU task.

Regardless of the fact that there are some differences between hierarchical SLU and NER, we can still apply the well-known techniques used in NER to an SLU problem. Following (Ramshaw and Marcus, 1995), the slot labels are drawn from a set of classes constructed by extending each label by X-B, X-I or O. Here, X-B means "begin a phrase of slot X", X-I means "continue a phrase of slot X" and O means "not in a phrase". For example, a phrase "new york" would be labeled as: "new/TOLOC.CITY_NAME-B york/TOLOC.CITY_NAME-I". In addition, a two-level hierarchical slot can be considered as an integrated flattened slot. For example, FROMLOC.CITY_NAME and TOLOC.CITY_NAME are different on this slot definition scheme.

In the statistical framework, the SLU problem can be stated as a sequential supervised learning problem (Dietterich, 2002). Let $\mathscr{D} = \{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}_{i=1,\ldots,N}$ be a set of N training examples. Each example is a pair of sequence $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$, where feature *vector sequence* $\mathbf{x}^{(i)} = \langle \mathbf{x}_1^{(i)}, \mathbf{x}_2^{(i)}, \ldots, \mathbf{x}_{T_i}^{(i)} \rangle$ and B/I/O *label sequence* $\mathbf{y}^{(i)} = \langle y_1^{(i)}, y_2^{(i)}, \ldots, y_{T_i}^{(i)} \rangle$ co-exist. The goal of statistical SLU is to construct a classifier h that can correctly predict a new label sequence given an input sequence $\mathbf{x}$, i.e., the goal of classifier h is to find the best probable semantic class sequence $\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} h(\mathbf{y}, \mathbf{x}, \Lambda)$.

Note that input sequence $\mathbf{x}$ and output sequence $\mathbf{y}$ are significantly correlated by a complex structure. In the sequential supervised learning task, the structure is simplified to left-to-right sequential structure. The family of techniques for solving such structured problems is generally known as structured prediction or structured learning in machine learning community. To date, there are several state-of-the-art structured learning algorithms such as hidden Markov model (HMM) (Rabiner, 1989), conditional random fields (Lafferty et al., 2001), maximum-margin Markov network (Taskar et al., 2003), support vector machine for structured outputs (Tsochantaridis et al., 2004) and search-based structured prediction (Daumé, 2006). The developer has a great freedom to design the function $h(\mathbf{y}, \mathbf{x}, \Lambda)$ using any structured prediction method.

The traditional way to build sequential model $h(\mathbf{y}, \mathbf{x}, \Lambda)$ is to use an HMM which represents the joint probability $P(\mathbf{y}, \mathbf{x})$. Using Bayes' rule, we specify an HMM using two probability distributions: transition distribution $P(y_t|y_{t-1})$ and observation distribution $P(\mathbf{x}_t|y_t)$. That is, the joint probability of a label sequence $\mathbf{y}$ and an observation sequence $\mathbf{x}$ factorizes as

$$P(\mathbf{y}, \mathbf{x}) = \prod_{t=1}^{T} P(y_t|y_{t-1}) P(\mathbf{x}_t|y_t). \tag{1}$$

Fig. 1. An example of a long-distance dependency problem in spoken language understanding. In this case, a word token "dec." with local feature set (dotted line box) is ambiguous for determining the correct label(DEPART.MONTH or RETURN.MONTH).

HMMs have been used for various sequential labeling tasks such as part-of-speech tagging and information extraction. However, modeling the joint distribution is difficult for exploiting the rich local features, because it requires modeling the complex dependencies to estimate distribution $P(\mathbf{x})$. The alternative is a discriminative approach that directly models the conditional distribution $P(\mathbf{y}|\mathbf{x})$. Using the discriminative classifier provides great flexibility to include a wide variety of arbitrary, non-independent features of the input.

For our statistical SLU system, we use a linear-chain CRF model; a model that assigns a joint probability distribution over labels which are conditional on the input sequences, where the distribution respects the independent relations encoded in a graph (Lafferty et al., 2001). A linear-chain CRF is defined as follows. Let $\mathcal{G}$ be an undirected model over sets of random variables $\mathbf{x}$ and $\mathbf{y}$. The graph $\mathcal{G}$ with parameters $\Lambda = \{\lambda_1, \ldots, \lambda_K\}$ defines a conditional probability for a state (or label) sequence $\mathbf{y}$ to be

$$P_\Lambda(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \cdot \exp\left(\sum_{t=1}^{T}\sum_{k=1}^{K} \lambda_k f_k(y_{t-1}, y_t, \mathbf{x}_t)\right), \tag{2}$$

where $Z(\mathbf{x})$ is the normalization factor that makes the probability of all state sequences sum to one.[1] In general, $f_k(y_{t-1}, y_t, \mathbf{x}_t)$ is an arbitrary linguistic feature function which is often binary-valued in NLP tasks. λ_k is a trained parameter associated with feature f_k. In our implementation, the feature function $f_k(y_{t-1}, y_t, \mathbf{x}_t)$ is factorized as $f_k^{\text{state}}(y_{t-1}, y_t)$ and $f_k^{\text{obs}}(y_t, \mathbf{x}_t)$. Then, the feature functions can encode any aspect of a state transition, $f_k^{\text{state}}(y_{t-1}, y_t)$, and the observation, $f_k^{\text{obs}}(y_t, \mathbf{x}_t)$, centered at the current time, t. Note that $f_k(y_t, \mathbf{x}_t)$ is state-dependent, and nonzero only for a single class. For example, a feature function $f_k(y_t, \mathbf{x}_t)$ could be a binary test that has value 1 if and only if $y_t =$ 'TOLOC.CITY_NAME-B' and current word is 'chicago'. Large positive values for λ_k indicate a preference for such an event, while large negative values make the event unlikely.

Parameter estimation of a linear-chain CRF is typically performed using conditional maximum log-likelihood (Lafferty et al., 2001; Sutton and McCallum, 2006). To avoid over fitting, the 2-norm regularization is applied to penalize on weight vector whose norm is too large. Then the penalized log-likelihood is

$$\mathcal{L}(\Lambda) = \sum_{i=1}^{N} \log P_\Lambda(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}) - \sum_{k=1}^{K}\frac{\lambda_k^2}{2\sigma^2} = \sum_{i=1}^{N}\sum_{t=1}^{T_i}\sum_{k=1}^{K} \lambda_k f_k(y_t^{(i)}, y_{t-1}^{(i)}, \mathbf{x}_t^{(i)}) - \sum_{i=1}^{N} \log Z(\mathbf{x}^{(i)}) - \sum_{k=1}^{K}\frac{\lambda_k^2}{2\sigma^2}. \tag{3}$$

We used a limited memory version of the quasi-Newton method (L-BFGS) (Nocedal and Wright, 1999) to optimize this objective function. The L-BFGS method converges super-linearly to the solution, hence it can be an efficient optimization technique on large-scale NLP problems (Malouf, 2002; Sha and Pereira, 2003).

A linear-chain CRF has been previously applied to obtain promising results in various natural language tasks (Pinto et al., 2003; Sha and Pereira, 2003; Peng and McCallum, 2004; McDonald and Pereira, 2005), but the linear-chain structure is deficient in modeling long-distance dependencies because of its limited structure (nth order Markov chains).

[1] This model is known as a HMM-like CRF (Sutton and McCallum, 2006). In this specialized model, a vector $\mathbf{x}_t$ includes context features (e.g. previous/next words, part-of-speech tags and phrase labels). For practicality, it is preferred to implement our SLU system. However, other types of linear-chain CRFs can be also used.

2.2. Long-distance dependency in spoken language understanding

In most sequential supervised learning problems including statistical SLU, the feature function $f_k(y_{t-1}, y_t, \mathbf{x}_t)$ indicates only local information for practical reasons. Although the state transition $f_k^{\text{state}}(y_{t-1}, y_t)$ is only dependent on a previous state, the observation feature vector $f_k^{\text{obs}}(y_t, \mathbf{x}_t)$ also uses local-based features. With sufficient local context (e.g. a sliding window of width 5) and first-order Markov chain structure, inference and learning are both efficient.

However, if we only use local features, then we cannot model long-distance dependencies. Thus, we should incorporate non-local information into the model. For example, Fig. 1 shows the long-distance dependency problem in an SLU task. The same two word tokens "dec." should be classified differently, `DEPART.MONTH` and `RETURN.MONTH`. The dotted line boxes represent local information at the current decision point ("dec."), but they are exactly the same in two distinct examples. Moreover, the two states share the same previous sequence (`O`, `O`, `FROMLOC.CITY_NAME-B`, `O`, `TOLOC.CITY_NAME-B`, `O`). If we cannot obtain higher-order dependencies such as "fly" and "return," then the linear-chain CRF cannot classify the correct labels between the two same tokens.

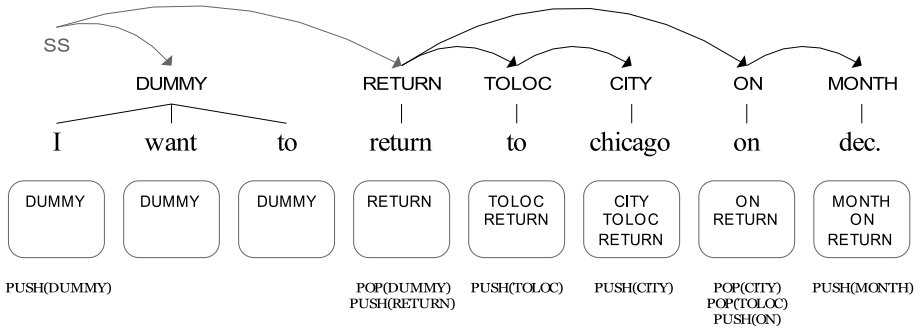
Some primary prior works propose solutions to the long-distance dependency for SLU task. He and Young (2005) describe an approach using hidden vector state (HVS) model which extends the basic HMM for encoding hierarchical structures. Fig. 2a shows an example of HVS, which describes (dependency parser-style) semantic parse tree and its vector state equivalent. Inference of HVS is based on a stochastic push-down automaton, i.e., state transitions are factored into a stack shift operation. This model can represent a right-branching context-free grammar (CFG) and can be robustly learned using EM algorithm (Dempster et al., 1977). However, the HVS model cannot exploit the rich and non-independent features, because it is a generative model. Moreover, EM learning is generally known as local-maxima solution.

Wang et al. (2005b) propose a simple approach to encoding the long-distance dependency. They build CRF-based SLU system for ATIS domain, and use previous slot context feature (PrevSlot) to capture non-local dependency. In Fig. 2b, for example, we assume that current decision point is "dec." state. To identify the label of "dec." the system uses preambles of previous slot "chicago." It is an effective and simple heuristic but the system requires domain-specific vocabulary lists or CFG-based rules to determine whether previous word is filler or preamble. Thus, it is difficult to apply to other domains or other sequential labeling tasks.

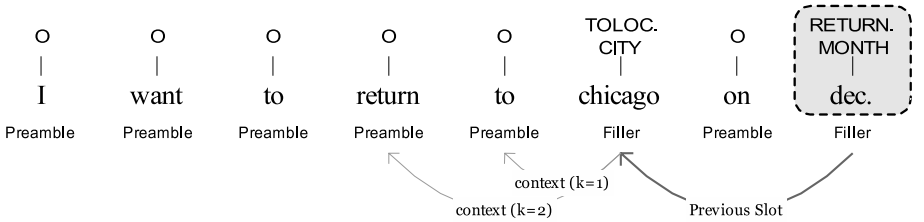
Schwartz et al. (1997) use the semantic parse trees to extract a meaning of sentence, but it is expensive to build the full-annotated parse tree data. Instead of domain-specific semantic parser, we can use domain-independent syntactic parse-tree information. Gildea and Jurafsky (2002) propose the parse tree path extracted from parse tree for semantic role labeling task. To apply this feature to SLU problem, we use a head-verb and a tree-path (Head/Path) because we assume that the most important dependency comes from the main-verb for SLU task. For example, Fig. 2c shows a tree path from target word "dec." to head verb "return." This approach is general and appropriate for capturing global dependency, but it requires a costly pre-processing and is impractical for real-time response in spoken dialog system. Moreover, the poor accuracy of the syntactic parser makes it difficult to exploit this type of feature. In addition, the recognized utterances are erroneous and the spoken language has no capital letters, no additional symbols, and sometimes no grammar, hence it is difficult to use a parser in an SLU problem.

Sutton and McCallum (2004) used a skip-chain CRF for information extraction from seminar announcement e-mail. One important kind of dependency within e-mail data occurs on repeated mentions of the same slots. When the same entity is mentioned more than once in a document, in many cases, all mentions have the same label. Adding edges called skip-chains allows to identically model long-range dependencies between repeated words. However, this approach cannot be directly applied to our SLU problem, because we hope to model the dependencies between arbitrary words rather than repeated words. Moreover, it is often intractable.

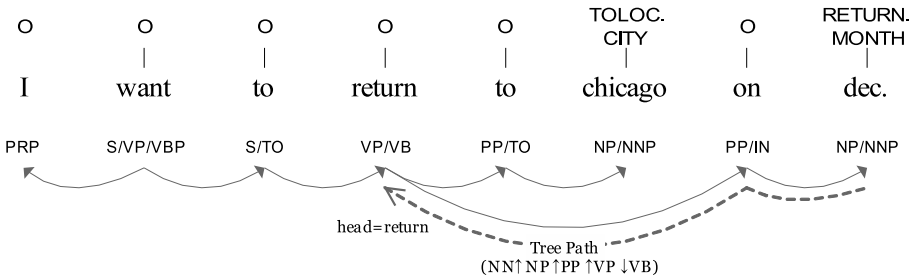
We summarize the previous works in terms of sequential labeling. HVS and skip-chain CRF explicitly incorporate non-local dependencies into sequential model, that is, a linear-chain structure is expanded to higher-order structure model. However, the computational costs of these models are exponentially increased by adding structures even a reasonable stack-depth restriction and approximate inference are introduced. PrevSlot and Head/Path approaches implicitly integrate the non-local dependencies, that is, we can employ



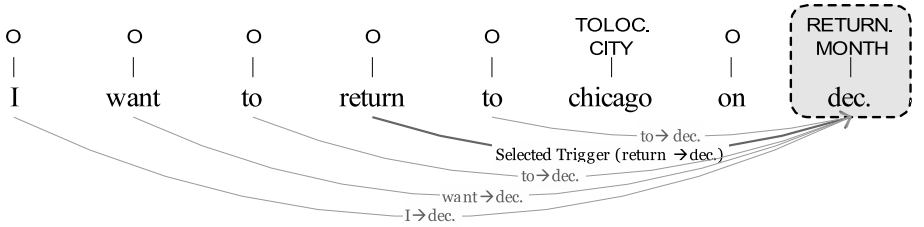
(a) Hidden vector state (He and Young, 2005)



(b) Previous slot's context (Wang et al., 2005b)



(c) Tree-path and head word (Gildea and Jurafsky, 2002)



(d) Trigger pair

Fig. 2. Approaches for long-distance dependency problem: (a) hidden vector state (He and Young, 2005); (b) previous slot's context (Wang et al., 2005b); (c) tree-path and head word (Gildea and Jurafsky, 2002) and (d) trigger pair.

the non-locality to an observational feature vector $f_k^{\text{obs}}(y_t, \mathbf{x}_t)$ without additional computational costs, which is more feasible for SLU task. However, a main disadvantage is that it is difficult to obtain these features because we need prior information such as the syntactic parse tree or domain-specific knowledge. To address this problem, we investigate a simple and general method to extract the non-local features for statistical SLU task.

Table 1
Examples of trigger features

Trigger type	Description	Example
$w_i \rightarrow w_j$, ($\|i-j\|>2$)	Word pair	return → dec.
$w_i \rightarrow \varepsilon$, ($\|i-j\|>2$)	Distinct unigram	return → ε
$f_i \rightarrow f_j$, ($\|i-j\|>2$)	Feature pair	return → CD
$f_i \rightarrow f_j$, ($i=j$)	Local conjunction	NNP → dec.
$\hat{y}_i \rightarrow \hat{y}_j$, ($\|i-j\|>2$)	Class label pair	RETURN.MONTH → RETURN.PERIOD

3. Incorporating non-local information

3.1. Using trigger features

In this section, we describe a data-driven method, which uses statistics to find features that are approximately modeling long-distance dependencies. The simplest way is to use identical words in history or lexical co-occurrence, but we wish to use a more general tool; triggering.

The trigger word pairs are introduced by Rosenfeld (1994). A trigger pair is the basic element for extracting information from the long-distance document history. In language modeling, *n*-gram based on the Markovian assumption cannot represent higher-order dependencies, but it can automatically extract trigger word pairs from data. The pair $w_i \rightarrow w_j$ means that words w_i and w_j are significantly correlated, that is, when w_i occurs in the document, it triggers w_j, causing its probability estimate to change. For example, Fig. 2d represents candidate trigger pairs and a selected pair "return → dec." for SLU task. In contrast to other methods, a trigger "return → dec." directly models non-local dependency between two words in a simple manner.

The trigger pairs previously introduced are just word pairs. Here, we can generalize the trigger word pairs to any arbitrary pairs of features. By extending word triggers, we define the general *trigger* features as follows.

Definition 1 (*Trigger pair*). Let two elements $a \in A$ and $b \in B$, A is a set of history features and B is a set of target features in training examples. If a feature element 'a' is significantly correlated with another element 'b', then $a \rightarrow b$ is considered as a **trigger**, with 'a' being the trigger element and 'b' the triggered element.

Some examples of triggers are listed in Table 1 and its graphical view is shown in Fig. 3.[2] Note that set B (target features) can be an empty set $\{\varepsilon\}$, which leads to null triggers, which can be interpreted by long-range unigram features.[3] It is useful to reduce the parameter size. For instance, some word triggers such as "return → dec.", "return → nov.", and "return → denver" can be generalized by "return → ε". In most natural language tasks, useful features include words, prefixes and suffixes, capitalization, part-of-speech tags and phrase chunk labels, membership in domain-specific lexicons, and semantic information from sources such as WordNet (Fellbaum, 1998). Following our definition, we can incorporate an arbitrary feature pair $f_i \rightarrow f_j$ into statistical SLU model. For example, the feature pair in/PP-B[4] → of is useful in deciding the correct answer PERIOD_OF_DAY-I in "in the middle *of* the day." Moreover, return → CD[5] effects to generalize the triggers for arbitrary day numbers. Without constraints on generating the pairs (e.g. at most three distant tokens, $|i-j|>2$), the candidates can be arbitrary conjunctions of features. "NNP → dec." is an example of local

[2] We define triggers as general pairs of two elements, 'a' and 'b.' Note that the element can not only be a single word but also a phrase or a conjunction. Moreover, they can include state variables, for example, '$b = x$ & y,' where 'x' is an arbitrary feature, 'y' is a state, and 'b' belongs to a set of cross-product space. In this paper, however, we regard trigger pairs as state-independent features, since the feature function in CRF model already represents the state-dependent observations.

[3] A word trigger pair is a general bigram between w_i and w_j where $i \neq j$. If one of two words is removed, then it is equivalent to the unigram feature (w_j or w_i). We call "$w_i \rightarrow \varepsilon$" null trigger, and it is a pseudo-pair trick where we uniformly represent all types of triggers.

[4] PP-B: begin a proposition phrase.

[5] CD: cardinal number.

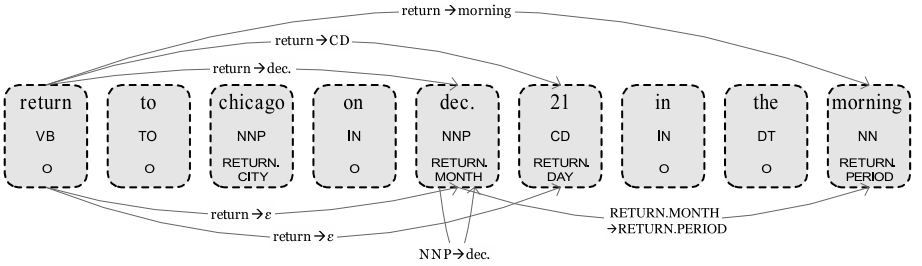


Fig. 3. Example of trigger features for air travel data.

conjunction feature ($i = j$) in Fig. 3. Moreover, we can explore the label–label pairs which represent higher-order dependencies between two hidden labels.[6] Therefore, we can explore any feature including local conjunction, non-local singleton, or structural dependency features in a uniform framework.

To select reasonable pairs from arbitrary pairs, Rosenfeld (1994) used average mutual information (MI). In this scheme, the MI score of one pair is calculated as follows:

$$MI(w_i;w_j) = P(w_i,w_j)\log\frac{P(w_j|w_i)}{P(w_j)} + P(w_i,\bar{w}_j)\log\frac{P(\bar{w}_j|w_i)}{P(\bar{w}_j)} + P(\bar{w}_i,w_j)\log\frac{P(w_j|\bar{w}_i)}{P(w_j)} + P(\bar{w}_i,\bar{w}_j)\log\frac{P(\bar{w}_j|\bar{w}_i)}{P(\bar{w}_j)}, \tag{4}$$

where $P(w_i,w_j)$ is a joint probability of words co-occurring within a context window and $P(w_i,\bar{w}_j)$ is a joint probability of word w_i occurring but not w_j. Note that it measures how far two words are from being independent in view of relative entropy. Using the MI criterion, we can select correlated word pairs. For example, the trigger pair "return → dec." was extracted with score 0.001179 in the training data, which can represent long-distance dependency and provide a cue to identify ambiguous classes. The MI approach, however, considers only lexical collocation without reference labels $\mathbf{y}$, and MI based selection tends to excessively select the irrelevant triggers. Recall that our goal is to find the significantly correlated trigger pairs which improve the model of classifier h in statistical SLU problems. Therefore, we need to use a more appropriate selection method for sequential supervised learning.

3.2. Selecting trigger features

We present another approach to extract relevant triggers and exploit them in sequential labeling. Our approach is based on an automatic feature induction (FI) algorithm, which is a novel method for selecting a feature in an exponential model (Della Pietra et al., 1997; McCallum, 2003). We follow McCallum's work, which is an efficient method for inducing features in a linear-chain CRF model. The CRF feature induction algorithm iteratively increases the bundle of mixed features including local features and trigger features. The algorithm is computationally costly even with the efficient calculation introduced. We therefore propose a simple but efficient modification to the algorithm, making it approximate but more feasible.

To select a relevant subset from a large collection of triggers, we begin by specifying the feature gain, which measures the effect of adding a feature. The feature gain is defined as the improvement in log-likelihood in training data.

Definition 2 (*Feature gain*). $\mathscr{L}^{\text{old}}(\Lambda)$ is the log-likelihood of previous model, and $\mathscr{L}^{\text{new}}(\Lambda,\mu)$ is the log-likelihood for a new model where μ is a parameter of an adding feature to be found and $g(y,\mathbf{x})$ is a corresponding feature function. Then, the feature gain is defined by a difference between new and old log-likelihood as $\Delta\mathscr{L}(\Lambda,\mu) = \mathscr{L}^{\text{new}}(\Lambda,\mu) - \mathscr{L}^{\text{old}}(\Lambda)$.

[6] This is similar to a consistency model to encode structural dependency in a skip-chain CRF (Sutton and McCallum, 2004). In contrast to the skip-chain model, we use previous classifier's prediction $\hat{\mathbf{y}}$ to create label–label pairs.

The feature induction algorithm greedily finds the subset of feature that improves the log-likelihood, i.e., our goal is to iteratively select a set of candidates which maximizes the gain $\Delta\mathscr{L}(\Lambda,\mu)$, and then adds it to the training data.[7] An algorithm can be summarized by two computations: calculating $\mathscr{L}^{\text{old}}(\Lambda)$ and $\mathscr{L}^{\text{new}}(\Lambda,\mu)$. Computing $\mathscr{L}^{\text{old}}(\Lambda)$ is equivalent to learning CRF model mentioned in Section 2.1. The remaining part is an estimation of the new log-likelihood $\mathscr{L}^{\text{new}}(\Lambda,\mu)$. Unfortunately, when a new feature $g(y,\mathbf{x})$ is included, the optimal values of all parameters Λ change. We therefore assume that the addition of a feature $g(y,\mathbf{x})$ affects only μ, leaving the other parameters unchanged. Then we can find the optimal parameter $\hat{\mu}$ to give maximum gain as in the following equation:

$$\Delta\mathscr{L}(\Lambda,\hat{\mu}) = \max_{\mu}\Delta\mathscr{L}(\Lambda,\mu) = \max_{\mu}(\mathscr{L}^{\text{new}}(\Lambda,\mu) - \mathscr{L}^{\text{old}}(\Lambda)). \tag{5}$$

The algorithm iteratively calculates the gain and selects the features with highest values. Each loop requires optimizing parameter μ. It is computationally costly because we need to optimize all candidate features which can be extremely large, e.g., Sha and Pereira (2003). Fortunately, our goal is to select *non-local* triggers, not local features. Therefore, we assume that a baseline model uses all local features and we only evaluate non-local features whether they are included or not.

Approximation 1 (*Baseline local model*). We assume that a baseline model should employ the whole local features. Instead of evaluating gains for all features, we only consider selecting of non-local triggers which improve performance of a baseline. This approximation is motivated by rule of thumb; completely use local information and selectively use global information.

Our basic assumption is that the local information should be included because the local features are the basis of the decision to identify the classes, and they reduce the mismatch between training and testing tasks. Furthermore, this assumption leads us to reduce iterations in the inducing procedure because we can only consider additional trigger features.

Following (Della Pietra et al., 1997; McCallum, 2003), the mean field approximation allows us to treat the calculation of $P_\Lambda(\mathbf{y}|\mathbf{x})$ as an independent inference problem rather than a sequential inference. That is, we approximate $P_\Lambda(\mathbf{y}|\mathbf{x})$ by the mean field $P_\Lambda(\mathbf{y}\mid\mathbf{x}) = \prod_t P_\Lambda(y_t\mid\mathbf{x}_t)$. It is the independent product of marginal distributions at each position t. We can calculate the marginal probability of state y with an adding trigger pair given observation $\mathbf{x}$ separately as $P_\Lambda(y_t|\mathbf{x}_t) = \alpha_t(y_t|\mathbf{x}_t)\cdot\ \beta_{t+1}(y_t|\mathbf{x}_t)/Z(\mathbf{x})$ by the definition of linear-chain CRF.

Although mean field approximation is efficient for estimating a new parameter μ, we still need to use the expensive forward–backward algorithm to compute the log-likelihood $\mathscr{L}^{\text{old}}(\Lambda)$ and to estimate marginal distribution $P_\Lambda(y_t|\mathbf{x}_t)$. Here, we introduce our second approximation. We encode non-local dependencies to observational features $f^{\text{obs}}(y_t,\mathbf{x}_t)$ not to the structure. Similar to approximation 1, we need not evaluate all the state transitions. We can take advantage of this fact by ignoring the transition feature $f^{\text{state}}(y_{t-1},y_t)$ to calculate a gain of the trigger feature.

Approximation 2 (*Sequential to independent model*). If we assume that all output variables $\mathbf{y}$ are independent, then we use the individual inference problem over the unstructured model whose state variable is independent from other states in history. That is, we approximate a linear-chain CRF to a conditional maximum entropy model in the trigger selection step.

This approximation allows us to estimate the change in log-likelihood independently for each position in t, which avoids the need for dynamic programming. The background of our approximation is that the state independent problem of CRF can be relaxed to maximum entropy (ME) inference problem without the state-structured model. In the result, we calculate the gain of candidate triggers, and select trigger features over a light

[7] Since the evaluation of relevant features is based on the log-likelihood, our selection algorithm should be state-dependent. While we did not consider the states in definition of trigger pairs, triggers are likely to be state-dependent in selection and learning steps. That is, the non-local feature function $g(y,\mathbf{x})$ is state-dependent, and triggers correlated with specific labels are likely to be more informative than other candidates in feature selection.

ME model instead of a huge computational CRF model. According to removal of the state transition, the marginal distribution of $P_\Lambda(y_t|\mathbf{x}_t)$ and a resulting gain are changed. Nevertheless, we assume that the absolute values of feature gain are different but relative ranks of candidate triggers are unchanged.

Formally, we approximate the log-likelihood $\mathscr{L}(\Lambda,\mu)$ as

$$\mathscr{L}(\Lambda,\mu)=\sum_{i=1}^{N}\log P_{\Lambda+\mu}(\mathbf{y}|\mathbf{x})\approx\sum_{i=1}^{N}\log\prod_{t=1}^{T_i}P_{\Lambda+\mu}(y_t|\mathbf{x}_t)=\sum_{i=1}^{N}\sum_{t=1}^{T_i}\log P_{\Lambda+\mu}(y_t|\mathbf{x}_t). \tag{6}$$

Note that $P_{\Lambda+\mu}(y_t|\mathbf{x}_t)$ is state-independent ME model and can be easily calculated. Since all training positions are independent, we can remove time index t, and Eq. 6 is replaced by $\sum_{i=1}^{M}\log P_{\Lambda+\mu}(y|\mathbf{x})$ where M is the total number of independent training samples ($=\sum_{i=1}^{N}\sum_{t=1}^{T_i}1$) that is equal to the number of word in training data. By adding a new candidate trigger, an additional feature model is as follows:

$$P_{\Lambda+\mu}(y|\mathbf{x})=\frac{\exp\left(\sum_k\lambda_k f_k(y,\mathbf{x})+\mu g(y,\mathbf{x})\right)}{\sum_y\exp\left(\sum_k\lambda_k f_k(y,\mathbf{x})+\mu g(y,\mathbf{x})\right)}=\frac{P_\Lambda(y|\mathbf{x})\exp(\mu g(y,\mathbf{x}))}{\sum_y P_\Lambda(y|\mathbf{x})\exp(\mu g(y,\mathbf{x}))}. \tag{7}$$

The gain of trigger features can be calculated on the old model that is trained with the local and added trigger pairs in previous iterations. With the candidate feature set, the gain is derived as:

$$\begin{aligned}\Delta\mathscr{L}(\Lambda,\mu)&=\underbrace{\sum_{i=1}^{M}\log(P_{\Lambda+\mu}(y^{(i)}|\mathbf{x}^{(i)}))-\frac{\mu^2}{2\sigma^2}-\sum_{k=1}^{K}\frac{\lambda_k^2}{2\sigma^2}}_{\mathscr{L}^{\text{new}}(g,\mu,\Lambda)}-\underbrace{\left(\sum_{i=1}^{M}\log(P_\Lambda(y^{(i)}|\mathbf{x}^{(i)}))-\sum_{k=1}^{K}\frac{\lambda_k^2}{2\sigma^2}\right)}_{\mathscr{L}^{\text{old}}(\Lambda)}\\&=\sum_{i=1}^{M}\log\left(\frac{\exp(\mu g(y^{(i)},\mathbf{x}^{(i)}))}{\sum_y P_\Lambda(y^{(i)}|\mathbf{x}^{(i)})\exp(\mu g(y^{(i)},\mathbf{x}^{(i)}))}\right)-\frac{\mu^2}{2\sigma^2}\\&=\sum_{i=1}^{M}\mu g(y^{(i)},\mathbf{x}^{(i)})-\sum_{i=1}^{M}\log\left(\sum_y P_\Lambda(y^{(i)}|\mathbf{x}^{(i)})\exp(\mu g(y^{(i)},\mathbf{x}^{(i)}))\right)-\frac{\mu^2}{2\sigma^2}.\end{aligned} \tag{8}$$

We can efficiently assess many candidate trigger features in parallel by assuming that the old features remain fixed while estimating the gain. The optimal value of $\hat{\mu}$ can be calculated by Newton's method. The first order derivatives are

$$\Delta\mathscr{L}'(\Lambda,\mu)=\sum_{i=1}^{M}g(y^{(i)},\mathbf{x}^{(i)})-\sum_{i=1}^{M}\sum_y P_{\Lambda+\mu}(y^{(i)}|\mathbf{x}^{(i)})g(y^{(i)},\mathbf{x}^{(i)})-\frac{\mu}{\sigma^2}. \tag{9}$$

And the second order derivatives are

$$\Delta\mathscr{L}''(\Lambda,\mu)=-\sum_{i=1}^{M}\sum_y P_{\Lambda+\mu}(y^{(i)}|\mathbf{x}^{(i)})\{g(y^{(i)},\mathbf{x}^{(i)})^2-g(y^{(i)},\mathbf{x}^{(i)})\sum_{y'}P_{\Lambda+\mu}(y'^{(i)}|\mathbf{x}^{(i)})g(y'^{(i)},\mathbf{x}^{(i)})\}-\frac{1}{\sigma^2}. \tag{10}$$

Finally, we can obtain the optimum parameter via Newton update:

$$\mu^{\text{new}}=\mu^{\text{old}}-\frac{\Delta\mathscr{L}'(\Lambda,\mu^{\text{old}})}{\Delta\mathscr{L}''(\Lambda,\mu^{\text{old}})}. \tag{11}$$

Recall that a non-local trigger function $g(y,\mathbf{x})$ depends on y. Hence, we need to sum over all output state y, because a trigger $a\rightarrow b$ should be y-independent for an unknown test event. Thus, the gain of $a\rightarrow b$ is calculated by

$$\text{Score}(a\rightarrow b)=\sum_y\Delta\mathscr{L}(\Lambda,\hat{\mu}). \tag{12}$$

Using Eq. (12), we can select a small portion of all candidates, and retrain the model with selected features. We iteratively perform the selection algorithm with some stop conditions (excess of maximum iteration or no added feature up to the gain threshold). Note that we only use independent ME model for feature induction procedure, and finally train the linear-chain CRF with selected triggers. The outline of the induction algorithm

is described in Algorithm 1. Note that the output of the algorithm is trigger-augmented training data, and it therefore leads us to flavor for choosing any structured prediction method.

Algorithm 1 Trigger selection

input: a threshold value δ, data $\mathscr{D}$
output: a set of trigger features t, trigger-augmented data $\mathscr{D}$
1: Initialize training data $\mathscr{D}$ with local features and a trigger set $t = \langle\rangle$
2: **while** t is increased
3: Learn a ME classifier on $\mathscr{D}$: $\Lambda \leftarrow \text{TrainME}(\mathscr{D})$
4: Make candidates: $g \leftarrow \text{GenerateTriggers}(\mathscr{D}, \Lambda)$
5: Optimize μ: $\hat{\mu} \leftarrow \text{OptimizeGain}(\mathscr{D}, g, \Lambda)$
6: Select triggers: $g^* \leftarrow \text{SelectTrigger}(g, \hat{\mu}, \delta)$
7: Update training data: $\mathscr{D} \leftarrow \text{UpdateData}(\mathscr{D}, g^*)$
8: Update a trigger set: $t \leftarrow t \cup g^*$
9: **end while**

3.3. Implementation issues

There are some implementation issues for our trigger selection algorithm. In the previous section, we introduced two approximations to reduce computation. Nevertheless, the total number of training positions can easily be huge (e.g. around 500,000 for our experiment). Thus, we should sum over all training positions to evaluate the log-likelihood and generate the candidate triggers from all training position. However, we can speed up by reducing the training position to be evaluated. We select a small amount of training positions M' where $M' \ll M$. There are several strategies:

- Strategy 1: using only misclassified examples $M' = \{i|\tilde{y}^{(i)} \neq \arg\max_y P_\Lambda(y^{(i)}|\mathbf{x}^{(i)})\}$ where $\tilde{y}^{(i)}$ is a true label
- Strategy 2: using low confident examples $M' = \{i|P_\Lambda(y^{(i)}|\mathbf{x}^{(i)}) < c\}$ where $0 \leqslant c \leqslant 1$ is a confidence
- Strategy 3: using random samples $M' = \{i|i \sim \mathit{uniform}(1, M)\}$

In our implementation, we used the strategy 1 introduced by McCallum (2003); rather than summing over all training instances, we only need to use the mislabeled M' tokens by the current parameter.

From misclassified instances, we generate the candidate triggers; all pairs of current position j and the others within a sentence (formally, $a \rightarrow b$, $\forall a \in \mathbf{x}$ and $\forall b \in \mathbf{x}_j$). In this case, a history element a is not only in left-side of j but also in right-side. Hence, we differentiate two triggers as $a_{\mathrm{L}} \rightarrow b$ for the left-side trigger and $a_{\mathrm{R}} \rightarrow b$ for the right-side trigger.[8]

If we extend the idea to other domains, which takes a document as input, generating all pairs within an input $\mathbf{x}$ is infeasible. For document-based system, the trigger pairs should be carefully generated by using some heuristics. There are several possibilities: (1) limit the maximum boundary of triggers, (2) prune the events infrequently occurred, (3) use phrase-based triggers (a, b or both are a sequence of word), and (4) exploit meta-information (e.g. html tag for web-document or headline for news article).

4. Experimental results

4.1. Air travel data and experimental setup

We evaluate our method on the CU-Communicator corpus (air travel) (Ward, 1990). It consists of almost 2300 dialogues (approximately 40,000 user utterances) in total. Following (He and Young, 2005), most of the single word responses, which have no content words (e.g. "Yes, please" or "No")

[8] In our experiment, the right-side triggers are extremely rare, because right-to-left dependency is unusual in spoken dialog. However, this representation could be useful for news article or web document.

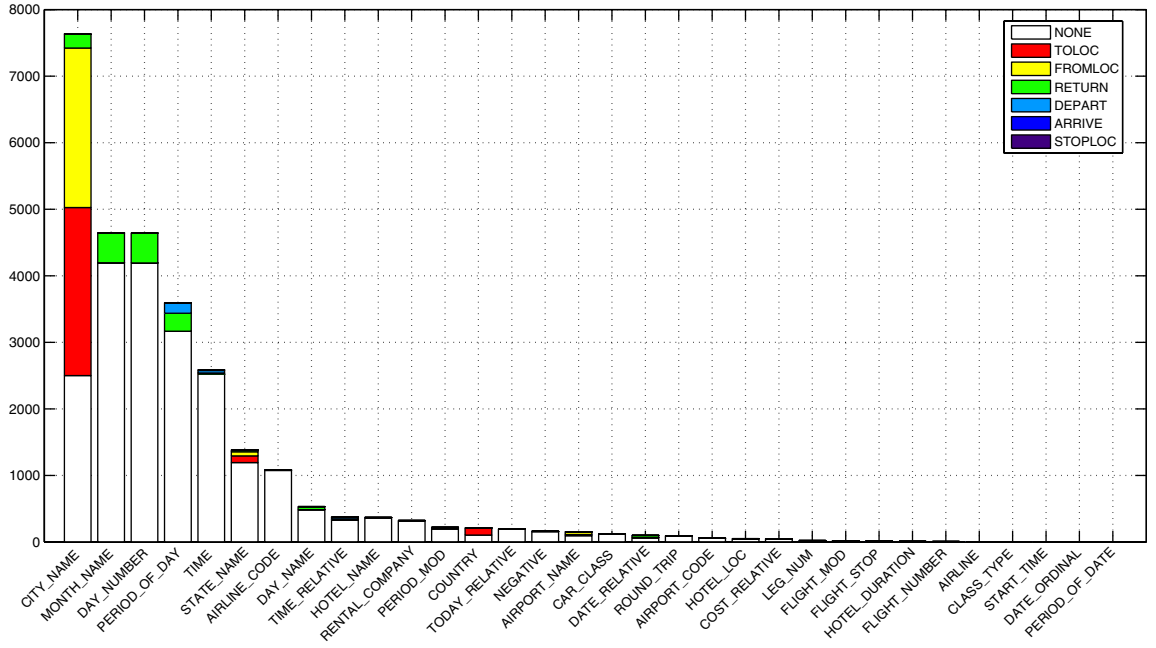


Fig. 4. A distribution of hierarchical classes in air travel corpus.

are removed and 13,983 utterances are used for our experiments. The semantic categories in this data set correspond to city names, time-related information, airlines and other miscellaneous entities. The semantic labels are automatically generated by a Phoenix parser (Pellom et al., 2000) and manually corrected. In the data set, the semantic category has a two-level hierarchy: 31 first level classes and 7 second level classes, for a total of 62 class combinations. Fig. 4 shows a distribution of hierarchical classes of this air travel data set. The data set is 630k words with 29k entities. Roughly half of the entities are time-related information, a quarter of the entities are city names, a tenth are state and country names, and a fifth are airline and airport names. For the second level hierarchy, approximately three quarters of the entities are `NONE`, a tenth are `TOLOC`, a tenth are `FROMLOC`, and the remainings are `RETURN`, `DEPERT`, `ARRIVE`, and `STOPLOC`. We evaluated the air travel set, and all of our results have averaged over 5-fold cross validation (CV) with an 80/20 split of the data.

For spoken inputs, we use the HTK-based recognition system (Young et al., 2005). We train the recognizer with only the domain-specific speech corpus (not only the acoustic model but also the language model). The reported accuracy for total air travel set is approximately 85% (Ward and Pellom, 1999), but the accuracy of our speech recognizer is 78.42%; we used only a subset of the data without tuning and the sentences of this subset are longer and more complex than the removed single-word responses. Next, we normalize the transcripts and speech recognition results. After pre-processing some disfluencies and digits (e.g. year, month, day number, and time), we can apply our method to both the text and the spoken inputs in a uniform manner.

As it is a standard, we compute precision and recall, which are evaluated on a per-entity basis and combined into a micro-averaged F_1 score ($F_1 = 2PR/(P + R)$). To verify the significance of our results, we use McNemar paired test (Gillick and Cox, 1989), which is based on individual labeling decision to compare correctness of two models. Refer to Sha and Pereira (2003) for more details.

A final model (a first-order linear chain CRF) is trained for 100 iterations with a Gaussian prior variance ($\sigma = 20$), and 200 or fewer trigger features (down to a gain threshold of 1.0) for each round of inducing iteration (100 iterations of L-BFGS for the ME inducer and 10–20 iterations of L-BFGS for the CRF inducer). All experiments are implemented in C++ and executed on Linux with XEON 2.8 GHz dual processors and 2.0 Gbyte of main memory.

4.2. Feature template

We list the feature templates used by our experiment for statistical SLU in Fig. 5. For local features, we use the indicators for specific words at location i, or locations within five words of i ($-2, -1, 0, +1, +2$ words on current position i). We also use the part-of-speech (POS) tags and phrase labels with partial parsing. Like words, the two basic linguistic features are located within five tokens. To compute feature performance, we begin with word features and iteratively add them one-by-one so that we can achieve the baseline performance. We separate out the local feature models as four types: only word feature (Local 1), word and POS-tag (Local 2), word and phrase chunk (Local 3), and all features (Local 4).

To compare with other approaches, we build two baseline non-local models using previous slot context (PrevSlot) and syntax parser-based (Head/Path) features. PrevSlot feature is originally designed by using rules to test whether previous history words are preamble or filler. However, it requires the domain knowledge such as entity lists or CFG-rules, hence this strategy cannot be directly compared to our method. Thus, we incorporate this type of features into CRF-based SLU model by using an alternative way as follows: the basic idea is to predict the labels twice in a cascaded manner (a two-step approach). First, we decide on a label sequence given a word sequence using an independent classifier (e.g. conditional ME classifier), and produce the PrevSlot features. Next, we perform the sequential labeling. In practice, however, using the true-labeled training data for first simple prediction step cannot be robust for new test data, because a noisy output, which does not appear in training data seriously, affects the result in this case. Alternatively, we used the predicted outputs of the first prediction in training step, and use CV technique (not required for test step). We reproduce the PrevSlot added training data by using 10-fold CV, and use it to estimate a CRF model of the system. This is an elaborate technique to reduce the training-test mismatch problem (Cohen and de Carvalho, 2005).

For non-local syntax parser-based features, we use Collins parser (Collins, 1999) and extract non-local type of features from the parse trees. The extracted features consist of the head word, POS-tag of the head word, head-verb word, and parse tree paths introduced by semantic role labeling task (Gildea and Jurafsky, 2002). Following the previous studies of semantic role labeling, the parse tree path improves the classification performance of semantic role labeling. For our experiment, we defined that a tree-path is a path from a target word to head-verb.

Finally, we use the trigger pairs that are automatically extracted from the training data. Avoiding the overlap of local features, we add the constraint $|i-j|>2$ (except Head/Path pairs). We extracted word triggers $w_i \rightarrow w_j$ from training data to model long-distance dependencies, and we add extended types of trigger. A null trigger is equivalent to a distinct unigram feature w_i outside of local context (see Fig. 3). We designed more sophisticated triggers motivated by previous works (Gildea and Jurafsky, 2002; Wang et al., 2005b). A label trigger is the pair between two labels which are produced by previous prediction (similar to PrevSlot and a

- Local feature templates
  - lexical words
  - part-of-speech (POS) tags
  - phrase chunk labels
- Previous slot context template
  - Previous slot word and its label
  - Previous slot context (previous two words)
- Head word and tree path feature template
  - head word/POS-tag and head verb word
  - parse tree path
- Trigger feature templates
  - word trigger $(w_i \rightarrow w_j)$, $|i-j|>2$
  - null trigger $(w_i \rightarrow \varepsilon)$, $|i-j|>2$
  - label trigger $(\hat{y}_i \rightarrow \hat{y}_j)$, $|i-j|>2$
  - Head/Path trigger $(h_i \rightarrow w_j)$, $i=j$

Fig. 5. Feature templates.

skip-chain CRF model). Note that the triggering slot $\hat{y}_i$ can be not only a previous slot but also one of all the slots. Basically, the trigger feature is a fully data-driven approach, but we can incorporate the external knowledge into our method. A Head/Path trigger uses syntactic parse tree information, which is a conjunction of Head/Path feature and word. While extended triggers increase the complexity of model, we expect that they would be helpful for capturing more complex dependencies.

4.3. Comparison of non-local methods

Tables 2 and 3 show the empirical results of local features, and three types of non-local models for text and spoken inputs, respectively. Using local models, we achieve best F_1 scores of 94.84 (Text) and 71.82 (ASR). The performance is slightly decreased by adding the additional local features (POS-tags and chunk labels) because the pre-processor brings more errors to the system. For each local model, we add the non-local features: PrevSlot, Head/Path, and word trigger. For both Text and ASR cases, the result shows that the trigger features are more robust to an SLU task than PrevSlot and Head/Path approaches. The trigger features significantly improve the performance of the statistical SLU. The differences between the trigger and the others are statistically significant (McNemar's test; $p < 0.001$).

The features[9] are 4227 ± 12.89 (Local1), 5540 ± 18.93 (Local2), 6557 ± 26.82 (Local3), and 7870 ± 33.55 (Local4), averaged 5-fold CV. These base local features are commonly used in non-local models. Training linear-chain CRFs takes approximately 1.5 hours, and there is no significant difference between local and non-local models. Therefore, we only measured pre-processing times to select non-local features. PrevSlot and Head/Path methods select 3190 ± 48.51 and 2069 ± 31.68 non-local features, respectively. Further, they take 402 ± 4.36 and 921 ± 5.50 seconds on average. They independently perform the feature selection for local models, thus all results are identical. In contrast, our method has selected a few non-local triggers; 608 ± 15.60, 612 ± 18.81, 546 ± 25.60, and 538 ± 30.36. Also, the trigger selection algorithm takes 330 ± 66.21, 333 ± 31.19, 339 ± 71.82, and 312 ± 53.91 seconds, respectively. Accordingly, our method can be an appealing way to exploit non-local features for spoken language applications.[10]

Figs. 6a and b show the difference in performance between the local and the non-local models as a function of training set size. The local model with the full training set of approximately 12,000 utterances performs equivalently to the trigger model on only 6000 training examples, a 50% reduction in the training set. In these figures, trigger-based model outperforms all other non-local models at all points.

Previous works on the air travel data have used 1178 test utterances with 18 classes for first level labels (He and Young, 2005). Table 4 compares our system to the previous results from (He and Young, 2005). We achieved 92.77 F_1-performance using the trigger features, as well as 39% of error reduction compared to the previous result.[11] The first and second rows show the results of the generative models, finite state transducer (FST) and hidden vector state (HVS). In contrast, our system uses a discriminative model, which directly models the conditional distribution, and is sufficient for classification task. To capture long-distance dependency, HVS uses a context-free-like model, which increases the complexity of the models. In contrast, we use non-local trigger features, which are relatively easy to use without having additional complexity of models.

4.4. Robustness for spoken inputs

To verify the effect of triggers for spoken inputs, we compare 1-best and N-best speech recognition results to the transcriptions. Fig. 7 presents the precision-recall curves for both spoken and text transcription inputs. Top-most two curves are obtained by using human transcriptions and the remaining curves are using ASR N-best and 1-best results. As we can see from this figure, improvements using trigger features are more signif-

[9] Since feature functions are state-dependent, actual parameters are much more. In general, it linearly increases in proportion to the number of output classes. In this paper, for brevity, we only report the number of features that are counts for unique observations in $\mathbf{x}$.

[10] Originally, PrevSlot using heuristic rules can reduce the cost of feature selection, but making reasonable rules are labor intensive. Similarly, using faster parser can reduce parsing time, but developing and adapting the parser is expensive.

[11] This test data was provided by Yulan He. She did not perform cross-validation, thus standard error is omitted in this result.

Table 2
Result of local and non-local models (Text)

	None	+PrevSlot	+Head/Path	+Trigger
Local 1	94.80 (±0.23)	95.23 (±0.22)	95.62 (±0.20)	**96.19** (±0.25)
Local 2	94.84 (±0.26)	95.51 (±0.30)	95.81 (±0.34)	**96.27** (±0.27)
Local 3	94.61 (±0.29)	95.50 (±0.31)	95.63 (±0.20)	**95.96** (±0.14)
Local 4	94.71 (±0.17)	95.44 (±0.31)	95.87 (±0.32)	**96.11** (±0.31)

F_1 scores are averaged over 5-fold CV's with standard errors.

Table 3
Result of local and non-local models (ASR)

	None	+PrevSlot	+Head/Path	+Trigger
Local 1	71.82 (±0.87)	72.06 (±0.75)	72.42 (±0.80)	**73.01** (±0.86)
Local 2	71.79 (±0.95)	72.04 (±0.77)	72.48 (±0.82)	**73.08** (±0.77)
Local 3	71.71 (±0.82)	72.12 (±0.74)	72.63 (±0.81)	**72.83** (±0.72)
Local 4	71.72 (±0.78)	72.13 (±0.78)	72.66 (±0.96)	**72.94** (±0.75)

F_1 scores are averaged over 5-fold CV's with standard errors.

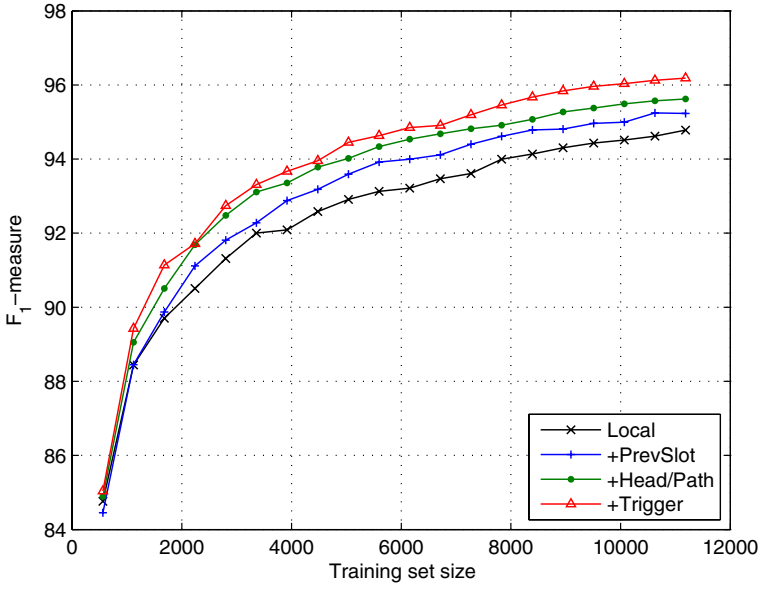
icant than those using N-best rescoring. We believe that our method efficiently extends to word confusion networks or lattices, and improves the performance by using lattice information (Hakkani-Tur et al., 2006).

4.5. Comparison of trigger selection methods

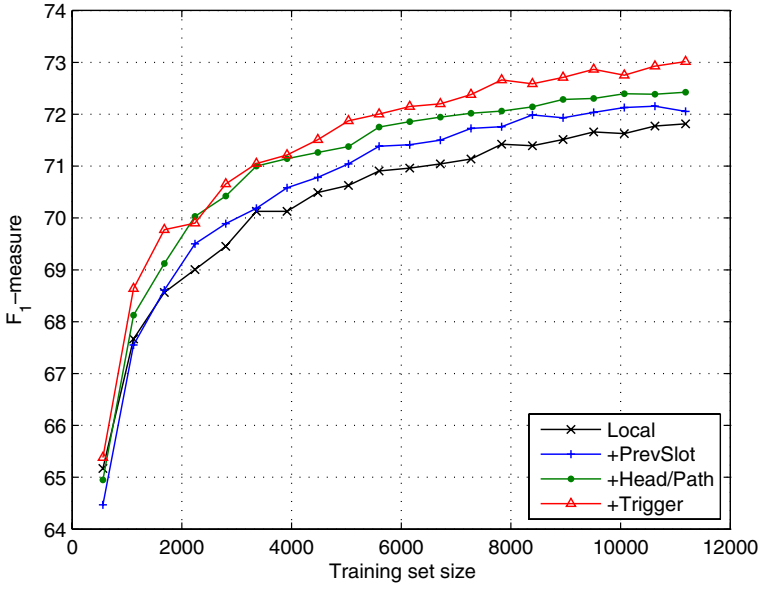
Next, we compare the two trigger selection methods: average mutual information (MI) and our feature induction (FI) (Section 3.2). Table 5 shows the experimental results of the comparison between MI and FI approaches. For the MI-based approach, we should calculate an average MI for each word pair appearing in a sentence and cut the unreliable pairs (down to threshold of 0.0001) before training the model. In contrast, the FI-based approach selects reliable triggers, which should improve the log-likelihood of the model. Our method based on the feature induction algorithm outperforms the simple MI-based methods (with statistical significance $p < 0.001$). Fewer features are selected by FI, that is, our method successfully prunes the event pairs that are highly correlated but not relevant to the models. The additional null, label, and Head/Path trigger features improve the performance over word trigger pairs, but they are less statistically significant to word trigger (word trigger vs. word + null trigger; $p = 0.610$, word + label trigger; $p = 0.159$, word + Head/Path trigger; $p = 0.005$). Nevertheless, adding null triggers are effective in reducing the size of the trigger features (608 to 285), because it has a generalization effect (e.g. "return $\rightarrow \varepsilon$" generalizes "return $\rightarrow$ dec.", "return $\rightarrow$ nov.", or "return $\rightarrow$ denver"). Although more trigger features are selected, label and Head/Path information improve the performance. There is a trade-off between model complexity and the performance to design and select the trigger-based non-local features.

Fig. 8 shows top 20 samples of triggers selected by MI and FI approaches. For example, the useful trigger "return $\rightarrow$ morning" is ranked in the first of FI but 66th of MI. Moreover, the top 8 pairs of MI are not meaningful, that is, MI selects many functional word pairs. The MI approach considers only lexical collocation without reference labels, hence the FI method is more appropriate to sequential supervised learning. Furthermore, the graph in Fig. 9 illustrates the difference in performance between FI and MI-based trigger selection methods as a function of trigger feature size. The FI-based selection dominates the MI-based method.

Our trigger induction algorithm is an iterative procedure to select high-ranked trigger pairs and to reject low-valued pairs. The rejecting decision is based on the user-control parameter, i.e., gain threshold, which dominates trigger size and performance. In iterations of induction algorithm, a large threshold value selects few features that affect the model in terms of estimated gains of candidate features. Fig. 10 illustrates the relation among threshold values, performance and feature size. Our feature induction method needs to be tuned for optimal threshold values. It appears that a systematic tuning method based on held-out data is necessary in the future.



(a) Text



(b) ASR

Fig. 6. Learning curves for the air travel data set, averaged over 5-fold CV's. Trigger model performs equivalently to the local model with two times more data: (a) text and (b) ASR.

4.6. *Effectiveness of selection algorithm*

Finally, we wish to justify that our modified version of the feature inducing algorithm is efficient and maintains performance without any drawbacks. We proposed two approximations: using whole local features (approx. 1) and using an unstructured model on the selection stage (approx. 2) (Section 3.2). Table 6 shows the results of variant versions of the algorithm. The first row of the table shows result of original CRF feature induction algorithm that has no additional approximation. Second and third rows are used by approx. 1 and

Table 4
Comparison of performance on air travel data (1178 test utterances used in He and Young (2005))

System	Precision	Recall	F_1 score
He & Young (FST)	82.34	82.94	82.64
He & Young (HVS)	88.84	87.31	88.07
Our system (CRF, Local)	91.86	91.17	91.52
Our system (CRF, +Trigger)	**92.98**	**92.56**	**92.77**

The trigger features outperform others. (The first two lines of this table are adapted from He and Young (2005).)

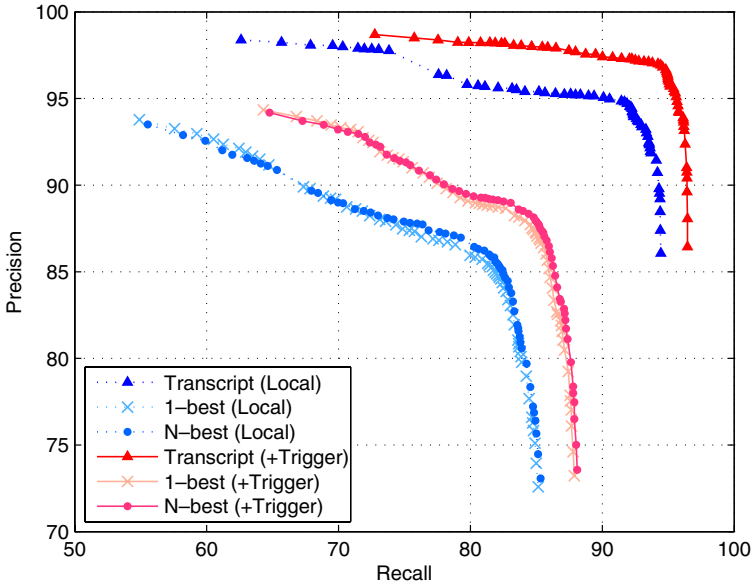


Fig. 7. A precision-recall curve that summarizes the experimental results on the air-travel data set.

Table 5
Result of the trigger selection methods

Method	# Triggers	Precision	Recall	F_1 score
Average mutual information				
Word trigger	1,713 (±11.9)	95.24 (±0.35)	95.24 (±0.28)	95.24 (±0.30)
Feature induction				
Word trigger	608 (±17.4)	96.20 (±0.26)	96.17 (±0.26)	96.19 (±0.25)
+null	**285** (±7.2)	96.22 (±0.23)	96.15 (±0.35)	96.18 (±0.24)
+label	735 (±36.8)	96.28 (±0.22)	96.31 (±0.21)	96.30 (±0.19)
+Head/Path	621 (±17.8)	**96.44** (±0.30)	**96.39** (±0.25)	**96.41** (±0.24)

approx. 2, respectively. And the last row is the result of the modified algorithm which uses both approx. 1 and 2. Surprisingly, the selection criterion based on ME (the unstructured model) is better than CRF (the structured model) not only for time cost but also for the performance on our experiment.[12] In addition, this result shows that local information provides the fundamental decision clues. Our modification of the algorithm to induce features for CRF is sufficiently fast for practical usage.

[12] In our analysis, 10–20 iterations for each round of inducing procedure are insufficient for optimizing the model in CRF inducer. Thus, the resulting parameters are under-fitted and selected features are infeasible. We need more iterations to fit the parameters, but they require too much learning time (>1 day).

Average Mutual Information (word)
like→from, i→to, to→on, from→on, i→from, i'd→from, like→on, i'd→fly, to→denver, on→in, to→and, on→and, i→go, december→ninety, denver→on, on→the, in→and, return→in, i→on, o'clock→morning
Feature Induction (word)
return→morning, on→morning, to→morning, on→afternoon, like→morning, return→afternoon, i'd→morning, in→morning, and→morning, the→morning, morning→morning, i→morning, december→afternoon, to→afternoon, leaving→morning, twenty→morning, the→afternoon, december→morning, wanna→morning, and→afternoon
Feature Induction (word+null pair)
return→ε, to→ε, on→ε, and→ε, like→ε, return→morning, the→ε, i→ε, on→morning, in→ε, i'd→ε, to→morning, morning→ε, twenty→ε, from→ε, december→ε, leave→ε, on→afternoon, like→morning, on→ε
Feature Induction (word+label)
on→PERIOD-B, return→PERIOD-B, to→PERIOD-B, RETURN.MONTH-B→PERIOD-B, RETURN.DAY-B→PERIOD-B, return→morning, RETURN.MONTH-B→morning, like→PERIOD-B, on→morning, RETURN.DAY-B→morning, to→morning, i'd→PERIOD-B, in→PERIOD-B, the→PERIOD-B, and→PERIOD-B, morning→PERIOD-B, DEPART.PERIOD-B→PERIOD-B, december→PERIOD-B, i→PERIOD-B, twenty→PERIOD-B
Feature Induction (word+Head/Path)
return→morning, head=return→morning, on→morning, head_tag=IN→morning, head=in→morning, return→afternoon, path=NN↑NP↑PP↓IN→morning, on→afternoon, morning→morning, and→morning, the→morning, head=return→afternoon, leaving→morning, to→morning, path=NN↑NP↑PP↑VP↓VB→morning, head=leaving→morning, return→twenty, head_tag=IN→afternoon, head=in→afternoon, path=NN↑NP↑PP↓IN→afternoon

Fig. 8. Top-20 trigger samples extracted by the various selection methods.

4.7. Evaluation for non-hierarchical SLU tasks

We applied our method to two Korean dialog data: telebanking and tv-epg (television electronic program guide). These two data sets are collected and annotated for developing spoken dialog systems and consist of non-hierarchical classes: (`about`, `amount`, `benefit`, `count`, `doing`, `info`, `kind`, `limit`, `period`, `qualified`, `rate`, `refund`, `rule`, `service`, `time` and `via`) for telebanking set, and (`anaphora`, `cast`, `channel`, `date`, `genre`, `program` and `time`) for tv-epg set. Some statistics of the two data sets are presented in Table 7. In practice, realistic dialog systems for human–computer interface prefer short and restrictive dialogs. The average number of words per utterance is less than 7–8 words and the average numbers of classes per an utterance is nearly 1–2 in these data sets. Table 8 presents the performance on two dialog data

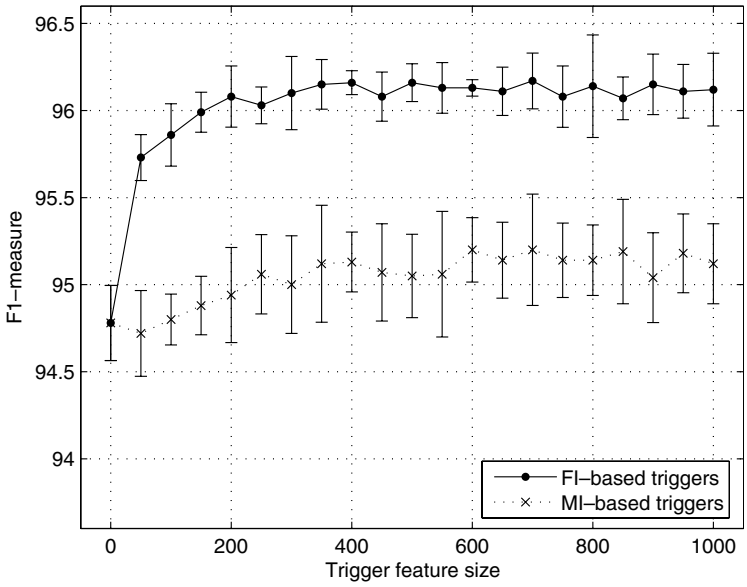


Fig. 9. Learning curves for the FI vs MI based trigger selection methods, averaged over 5-fold CV's. Each error bar represents a standard error of the results for 5-fold CV's.

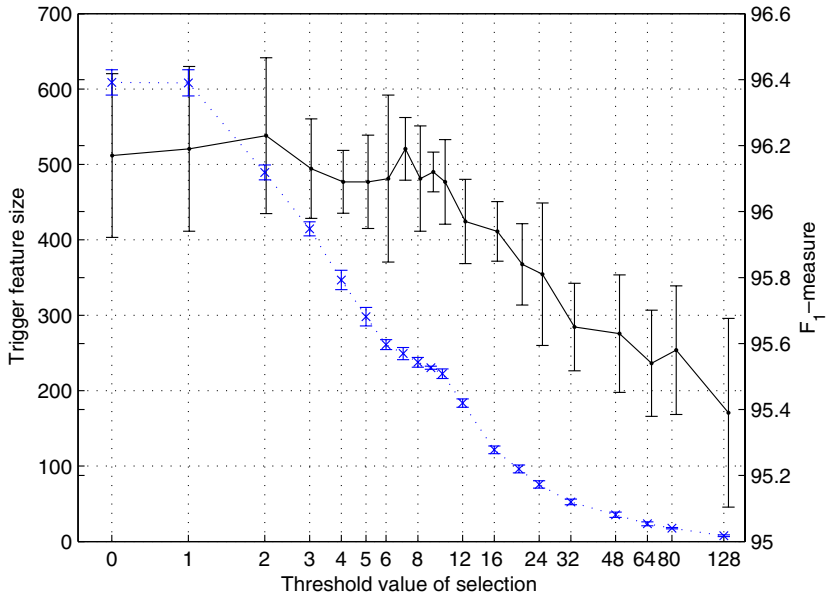


Fig. 10. Effect of gain threshold values.

Table 6
Comparison of variations in the induction algorithm

Inducer type	Time		# Features		F_1 score	
	Inducing	Training	Trigger	Local	Text	ASR
None	3:47:43	5:12:11	1,320	2,050	88.36	66.88
Approx. 1	0:47:30	2:14:02	800	4,227	95.17	72.23
Approx. 2	0:11:10	1:34:37	**467**	**1,421**	95.87	72.72
Approx. 1+2	**0:5:30**	**1:29:26**	608	4,227	**96.19**	**73.01**

Columns are induction and total training time (h:m:s), number of trigger and local features, and F_1-score on test data.

Table 7
Statistics of two spoken dialog data

Data set	# Utterance (utt)	# Unique class	# Word/utt	# Class/utt
Telebanking	2239	17	7.36	1.8
tv-epg	1917	7	5.15	1.2

Table 8
SLU results of two spoken dialog data

Data set	Model	Precision	Recall	F_1 score
Telebanking	Local	89.21 (±1.41)	87.92 (±1.74)	88.55 (±1.48)
	+Trigger	**90.54** (±1.43)	**88.92** (±1.99)	**89.72** (±1.68)
tv-epg	Local	96.84 (±1.16)	95.75 (±0.87)	96.29 (±0.99)
	+Trigger	**97.05** (±0.96)	**95.83** (±0.70)	**96.44** (±0.79)

sets, averaged on 5-fold CV's. The non-local trigger features are also valuable in predicting short and non-hierarchical classification problems (McNemar's test; $p = 0.01$).

5. Related works and discussions

The language understanding is mainly considered by many NLP tasks (MUC7, 1998), and most methods are based on rich linguistic information such as syntactic parse tree or lexicon-type knowledge. However, the methodologies invented on text are sometimes fragile when connected to a speech recognizer, because spoken language loses much information including grammar structure or morphology and causes many errors in automatically recognized speech. Understanding spoken language has been studied extensively from the latter half of the 1980. The basic roles of SLU are to analyze the output of the speech recognizer and to assign a meaning representation that can be used by the dialog manager. Both knowledge-based and statistical systems were developed for the SLU tasks.

Knowledge-based SLU systems use a large set of rules to explain the syntactic and semantic possibilities for spoken utterances, but the systems lack robustness when faced with the wide variety of ill-formed spoken inputs. The systems have typically been implemented via hand-crafted semantic grammar rules using domain-specific semantic parsers such as MIT's TINA (Seneff, 1992), CMU's PHOENIX (Ward, 1990) and SRI's Gemini (Dowding et al., 1993). However, a knowledge-based system is expensive to develop for several reasons: (1) combined linguistic and engineering expertise is required to construct or to maintain grammar rules with good coverage and optimized performance; (2) it is difficult to adapt or expand to other domains; and (3) it can lack robustness when error rates rise or unexpected syntactic forms are used.

SLU approaches based on statistical learning directly address many of the problems associated with the knowledge-based system. Statistical SLU methods directly map from word strings to intended meaning structures. In statistical framework, statistical models can be automatically learned from training examples. Compared to the manual grammar construction, labeling semantic classes for training data is easier to create and maintain, and requires less expert knowledge. Moreover, modern machine learning methods can be used to adapt to new data. Thus, statistical framework is a much more feasible solution for developing SLU system. Statistical methods for SLU have been investigated in the AT&T's CHRONUS (Levin and Pieraccini, 1995) and BBN's hidden understanding model (Schwartz et al., 1997). These models were motivated by HMM which has been extremely successful in both speech recognition and NLP.

The most relevant previous work in statistical SLU is He and Young (2005) which describes an approach using extended HMM, resulting in a model with some of the power of context-free models. This approach is competitive with our method, in the sense of the generative versus discriminative (conditional) model, and the representing dependency with the hierarchical model versus non-local features. However, Wang et al. (2005b) reported that the conditional model reduced SLU slot error rate by 17% over the generative HMM/CFG com-

posite model on ATIS data. Similarly, our approach performs significantly better than the generative approach.

More recently, some approaches have been developed to improve performance and adaptability for statistical SLU. The output of speech recognizer contains rich information, such as *N*-best list, word lattice and confidence scores, and many researchers made an effort for utilizing those kinds of information to improve SLU performance as well as providing useful information for dialog management. Raymond et al. (2006) tightly integrated SLU with ASR using finite state transducers, in which the probability of interpretation errors can be reduced. Hakkani-Tur et al. (2006) addressed robustness issues to ASR errors by using word confusion networks, which are compact representations of word lattices. We believe that our approach can be improved by combining fluent information such as word confusion networks or acoustic confidence scores with non-local information.

Several works have successfully incorporated non-local information into the probabilistic sequence model. In an NER task, Bunescu and Mooney (2004) used a relational Markov network (RMN) (Taskar et al., 2002) to explicitly model long-distance dependencies, but a disadvantage of this approach is that the dependencies must be defined manually. Sutton and McCallum (2004) introduced a skip-chain CRF which extends a linear-chain CRF by adding probabilistic connections between similar words, and it attempts to identify all mentions of an entity. By adding edges between similar words, the skip-chain CRF can represent dependencies between distant occurrences. Finkel et al. (2005) presented a method to use a non-local entity using Gibbs sampling for inference in a consistency model. However, those previous works only consider the relationship between similar words (e.g. same lexical words) and require additional constraints in inference time. By contrast, our method is based on features rather than structure (but the trigger features can be applied to an arbitrary structured CRF model).

The relevant prior work in an NER task is from Chieu and Ng (2002), who use additional features taken from other occurrences of the same token in the document. This approach has the advantage of allowing the training to automatically learn good weightings for these *global* features relative to the local ones. To exploit dictionaries in NER task, Sarawagi and Cohen (2004) suggested a semi-Markov CRF, in which the state transition probabilities depend on how long the chain has been in its current state. However, this type of "global" features must be defined and encoded by pre-compiled list, but our trigger features are automatically found from given data in a training procedure.

Trigger word pairs are introduced and successfully applied in a language modeling task. Rosenfeld (1994) demonstrated that the trigger word pairs improve the perplexity in ME-based language models. Our method extends this idea to sequential supervised learning problems. Our trigger selection criterion is based on the automatic feature inducing algorithm, and it allows us to generalize the method to the arbitrary pairs of features.

Our method is based on two previous works of feature induction on an exponential model (Della Pietra et al., 1997; McCallum, 2003). Our induction algorithm was built on the McCallum's method, which presents an efficient procedure to induce features on CRF. McCallum (2003) suggested using only the mislabeled events rather than the whole training events. This intuitional suggestion has offered us a fast training. We added two additional approximations to reduce the time cost: (1) an inducing procedure over a conditional non-structured inference problem rather than an approximated sequential inference problem, and (2) training with a local feature set, which is the basic information to identify the labels.

6. Summary and future works

We have presented a method for exploiting non-local information into a sequential supervised learning task. In a real-world problem such as statistical SLU, our model performs significantly better than the other non-local models. By comparing our selection criterion, we find that the mutual information tends to excessively select the triggers while our feature induction algorithm alleviates this issue. Furthermore, the modified version of the algorithm is practically fast enough to maintain its performance particularly when the local features are offered by the starting position of the algorithm.

In this paper, we have focused on a sequential model such as a linear-chain CRF. However, our method can also be naturally applied to arbitrary structured models, thus the first alternative is to combine our methods

with a maximum-margin Markov network (Taskar et al., 2003), support vector machine for structured outputs (Tsochantaridis et al., 2004) or semi-Markov CRF (Sarawagi and Cohen, 2004). Applying and extending our approach to other natural language tasks (which are difficult to apply a parser to) such as information extraction from e-mail data or biomedical named entity recognition is a topic of our future works.

Acknowledgments

We thank anonymous reviewers for helpful comments, and Yulan He for making the CU-Communicator data available. This research was supported by the MIC (Ministry of Information and Communication), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Assessment) (IITA-2006-(C1090-0603-0045)).

References

Bunescu, R., Mooney, R., 2004. Collective information extraction with relational markov networks. In: Proceedings of the Conference of the Association for Computational Linguistics (ACL), pp. 439–446.

Charniak, E., Johnson, M., 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In: Proceedings of the Conference of the Association for Computational Linguistics (ACL), Ann Arbor, MI, pp. 363–370.

Chieu, H.L., Ng, H.T., 2002. Named entity recognition: a maximum entropy approach using global features. In: Proceedings of the International Conference on Computational Linguistics (COLING), pp. 190–196.

Cohen, W.W., de Carvalho, V.R., 2005. Stacked sequential learning. In: Proceedings of the International Joint Conference of Artificial Intelligence (IJCAI), pp. 671–676.

Collins, M., 1999. Head-driven statistical models for natural language parsing. Technical Report, University of Pennsylvania.

DauméIII, H., 2006. Practical structured learning techniques for natural language processing. Technical Report, Los Angeles, CA.

Della Pietra, S., Della Pietra, V., Lafferty, J., 1997. Inducing features of random fields. IEEE Transactions on Pattern Analysis and Machine Intelligence 19 (4), 380–393.

Dempster, A.P., Laird, N.M., Rubin, D.B., 1977. Maximum likelihood from incomplete data via the em algorithm (with discussion). Journal of the Royal Statistical Society B 39, 1–38.

Dietterich, T.G., 2002. Machine learning for sequential data: a review. In: Caelli, T. (Ed.) Structural, Syntactic, and Statistical Pattern Recognition; Lecture Notes in Computer Science, vol. 2396, pp. 15–30.

Dowding, J., Gawron, J.M., Appelt, D.E., Bear, J., Cherny, L., Moore, R., Moran, D.B., 1993. Gemini: a natural language system for spoken-language understanding. In: Proceedings of the Conference of the Association for Computational Linguistics (ACL), pp. 54–61.

Fellbaum, C., 1998. Wordnet: An Electronic Lexical Database. MIT Press, Cambridge, MA.

Fine, S., Singer, Y., Tishby, N., 1998. The hierarchical hidden markov model: analysis and applications. Machine Learning 32 (1), 41–62.

Finkel, J.R., Grenager, T., Manning, C., 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In: Proceedings of the Conference of the Association for Computational Linguistics (ACL), Ann Arbor, MI, pp. 363–370.

Gildea, D., Jurafsky, D., 2002. Automatic labeling of semantic roles. Computational Linguistics 28 (3), 245–288.

Gillick, L., Cox, S., 1989. Some statistical issues in the comparison of speech recognition algorithms. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 532–535.

Hakkani-Tur, D., Bechet, F., Riccardi, G., Tur, G., 2006. Beyond asr 1-best: using word confusion networks in spoken language understanding. Computer Speech & Language 20 (4), 495–514.

He, Y., Young, S., 2005. Semantic processing using the hidden vector state model. Computer Speech & Language 19 (1), 85–106.

Jeong, M., Lee, G.G., 2006. Exploiting non-local features for spoken language understanding. In: Proceedings of the Joint International Conference on Computational Linguistics and Association of Computational Linguistics (COLING/ACL), Sydney, Australia, pp. 412–419.

Lafferty, J., McCallum, A., Pereira, F., 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proceedings of the International Conference on Machine Learning (ICML), pp. 282–289.

Levin, E., Pieraccini, R., 1995. Chronus, the next generation. In: Proceedings of ARPA Spoken Language Systems Technical Workshop, Austin, TX, pp. 269–271.

Malouf, R., 2002. A comparison of algorithms for maximum entropy parameter estimation. In: Proceedings of the Conference on Natural Language Learning (CoNLL), pp. 49–55.

McCallum, A., 2003. Efficiently inducing features of conditional random fields. In: Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI), p. 403.

McDonald, R., Pereira, F., 2005. Identifying gene and protein mentions in text using conditional random fields. BMC Bioinformatics 6 (Suppl. 1).

MUC7, 1998. In: Proceedings of the 7th Message Understanding Conference (muc-7). Available from: <http://www-nlpir.nist.gov/related_projects/muc/>.

Nocedal, J., Wright, S.J., 1999. Numerical Optimization. Springer, New York.

Peckham, J., 1991. Speech understanding and dialogue over the telephone: an overview of the esprit sundial project. In: DARPA Speech and Natural Language Workshop, Pacific Grove, CA.

Pellom, B.L., Ward, W., Pradhan, S. S., 2000. The CU communicator: an architecture for dialogue systems. In: Proceedings of the International Conference on Spoken Language Processing (ICSLP).

Peng, F., McCallum, A., 2004. Accurate information extraction from research papers using conditional random fields. In: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics and Human Language Technology (NAACL/HLT), pp. 329–336.

Pinto, D., McCallum, A., Lee, X., Croft, W., 2003. Table extraction using conditional random fields. In: Proceedings of the Conference on Research and Developments in Information Retrieval (SIGIR).

Price, P.J., 1990. Evaluation of spoken language systems: the atis domain. In: DARPA Speech and Natural Language Workshop, Hidden Valley, PA.

Rabiner, L.R., 1989. A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of the IEEE 77 (2), 257–286.

Ramshaw, L.A., Marcus, M.P., 1995. Text chunking using transformation-based learning. In: Proceedings of 3rd Workshop on Very Large Corpora, pp. 82–94.

Raymond, R., Bechet, F., Mori, R.D., Damnati, G., 2006. On the use of finite state transducers for semantic interpresentation. Speech Communication 48 (3–4), 288–304.

Rosenfeld, R., 1994. Adaptive statistical language modeling: a maximum entropy approach.

Sarawagi, S., Cohen, W., 2004. Semi-markov conditional random fields for information extraction. In: Proceedings of the International Conference on Machine Learning (ICML).

Schapire, R., Rochery, M., Rahim, M., Gupta, N., 2002. Incorporating prior knowledge into boosting. In: Proceedings of the International Conference on Machine Learning (ICML).

Schwartz, R., Miller, S., Stallard, S., Makhoul, J., 1997. Hidden understanding models for statistical sentence understanding. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Washington, DC, USA.

Seneff, S., 1992. Tina: a natural language system for spoken language applications. Computational Linguistics 18 (1), 61–86.

Sha, F., Pereira, F., 2003. Shallow parsing with conditional random fields. In: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics and Human Language Technology (NAACL/HLT).

Sutton, C., McCallum, A., 2004. Collective segmentation and labeling of distant entities in information extraction. In: Workshop on Statistical Relational Learning at ICML, Banff, Canada.

Sutton, C., McCallum, A., 2006. An Introduction to Conditional Random Fields for Relational Learning. MIT Press, Cambridge, MA.

Taskar, B., Abbeel, P., Koller, D., 2002. Discriminative probabilistic models for relational data. In: Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI), Edmonton, Canada, pp. 485–494.

Taskar, B., Guestrin, C., Koller, D., 2003. Max-margin markov networks. In: Proceedings of the Advances in Neural Information Processing Systems (NIPS).

Tsochantaridis, I., Hofmann, T., Joachims, T., Altun, Y., 2004. Support vector learning for interdependent and structured output spaces. In: Proceedings of the International Conference on Machine Learning (ICML).

Tur, G., Hakkani-Tur, D., Schapire, R.E., 2005. Combining active and semi-supervised learning for spoken language understanding. Speech Communication 45 (2), 171–186.

Walker, M., Rudnicky, A., Prasad, R., Aberdeen, J., Bratt, E., Garofolo, J., Hastie, H., Le, A., Pellom, B., Potamianos, A., Passonneau, R., Roukos, S., Sanders, G., Seneff, S., Stallard, D., 2002. Darpa communicator: Cross-system results for the 2001 evaluation. In: Proceedings of the International Conference on Spoken Language Processing (ICSLP).

Wang, Y., Deng, L., Acero, A., 2005a. Spoken language understanding: an introduction to the statistical framework. IEEE Signal Processing Magazine 22 (5), 16–31.

Wang, Y., Lee, J., Mahajan, M., Acero, A., December 2005b. Statistical spoken language understanding: from generative model to conditional model. In: Workshop on Advances in Structured Learning for Text and Speech Processing at NIPS.

Ward, W., 1990. The cmu air travel information service: understanding spontaneous speech. In: Proceedings of ARPA Spoken Language Systems Technical Workshop, pp. 127–129.

Ward, W., Pellom, B., 1999. The CU-communicator system. In: Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding.

Young, S., Evermann, G., Gales, M., Hain, T., Kershaw, D., Moore, G., Ollason, J.O. D., Povey, D., Valtchev, V., Woodland, P., 2005. The htk book: version 3.3. Technical Report, Cambridge University, UK. Avilable from: <http://htk.eng.cam.ac.uk/>.