DISTRIBUTED OPEN-DOMAIN CONVERSATIONAL UNDERSTANDING FRAMEWORK WITH DOMAIN INDEPENDENT EXTRACTORS

Qi Li[1,2] *Gokhan Tur*[1] *Dilek Hakkani-Tür*[1] *Xiang Li*[1,3] *Tim Paek*[1] *Asela Gunawardana*[1] *Chris Quirk*[1]

[1]Microsoft Research
[2]Rensselaer Polytechnic Institute
[3] New York University

ABSTRACT

Traditional spoken dialog systems are usually based on a centralized architecture, in which the number of domains is predefined, and the provider is fixed for a given domain and intent. The spoken language understanding (SLU) component is responsible for detecting domain and intents, and filling domain-specific slots. It is expensive and time-consuming in this architecture to add new and/or competing domains, intents, or providers. The rapid growth of service providers in the mobile computing market calls for an extensible dialog system framework. This paper presents a distributed dialog infrastructure where each domain or provider is agnostic of others, and processes the user utterances independently using their own knowledge or models, so that a new domain and new provider can be easily incorporated in. In addition, to facilitate each service provider building their own SLU models or algorithms, we introduce a new component, extractors, to provide intermediate semantic annotations such as entity mention tags, which can be plugged in arbitrarily as well. Each service provider can then rapidly develop their SLU parser with minimum efforts by providing some example sentences with intents and slots if needed. Our preliminary experimental results demonstrate the power of this new framework compared to a centralized architecture.

Index Terms— Distributed dialog system, domain detection, slot filling, entity extraction, spoken language understanding.

1. INTRODUCTION

Extending the coverage of traditional, centralized conversational interaction systems to new and/or competing domains, user intents and service providers is expensive and time-consuming. A single spoken language understanding (SLU) component is responsible for classifying domains and intents, and filling semantic slots. Although this design is simple, it is challenging and costly to develop a SLU system for multiple domains, given that it has to acquire and maintain expert knowledge and resources for all of them. Moreover, it is difficult to add new domains and service providers into this type of architecture. In recent years, the mobile and web computing has been growing fast. For example, there were 130k mobile apps in Windows Phone Store in February of 2013, this number increased to 255k by June of 2014 [1]. It has been increasingly important for a dialog system to rapidly incorporate emerging applications and service providers.

In this paper, we describe a distributed SLU framework, where multiple domains develop and perform the task of SLU independently, so that adding new domains and service providers does not require modification of existing ones. However, it is expensive and time-consuming for each individual domain or service provider to implement their own SLU models. On the other hand, multiple domains often share some common slots and semantic information. For example, locations are important in many domains such as *Weather* and *Restaurant*. [2] improved multi-domain slot filling by utilizing shared slots. To reduce the difficulty of each domain, leverage the shared information, as well as make the framework more flexible, we introduce a new component, domain-independent extractors, to extract various generic semantic information from speech utterances. Each domain can then build up its domain-specific SLU at low cost based on the available semantic annotations. Finally, the distributed framework also allows for benefiting from the active learning research such as [3, 4].

Most recent research on multi-domain SLU focused on a closed set of domains, such as [5, 6]. [7, 8] proposed a distributed agent architecture for multi-domain spoken dialog to reduce the complexity of centralized architecture. [9, 10] studied the domain switch problem for this distributed architecture. Although in their architectures different domains can be developed independently, they still need a master module to select domain for each utterance. Our work is also related to [11], which employed reusable SLU models to develop a dialog system quickly and with less supervision. The main contributions of this work are two-fold:

1. We present a distributed framework for multi-domain SLU that can incorporate new domains at low cost.
2. We abstract domain-independent SLU components from the implementation of each domain-specific SLU system

Domain	Example Intent	Example Slot Types
Weather	Check Weather	date, location
Places	Find Place	place name, location

Table 1: Example of intents and slot types.

by introducing a new and extensible component: domain-independent extractors. This enables a new domain or service provider to develop its SLU modules rapidly based on those intermediate components.

In our experiments (Section 4), we empirically analyze the impact of the proposed framework on the performance of SLU components.

2. SPOKEN LANGUAGE UNDERSTANDING

Spoken language understanding (SLU) aims at extracting meaningful information from speech utterances, so that the spoken dialog system can understand user request, and then take appropriate response. In the multi-domain scenario, a SLU module typically consists of *domain detection*, *intent detection*, and *semantic slot filling*.

The tasks of domain detection and intent determination are to classify a given speech utterance x into one of M semantic classes, $\hat{c}$, based on the contents of the utterance:

$$\hat{c} = \operatorname*{argmax}_{c \in M} p(c|x) \tag{1}$$

To this end, researchers have tried various classification methods such as Boosting [12, 13, 14], support vector machines (SVMs) [15], and more recently deep learning [16, 17]

In conventional SLU framework, M is a fixed number for both tasks. In reality, it has to be changeable.

The semantic structure of a domain is defined in terms of the semantic frames. Each semantic frame contains several typed components called "slots." The task of slot filling is then to instantiate the semantic frames from the speech utterance. In the simplest case, the semantic frame can be represented as tuples of ⟨slot-type, value⟩. Formally, the task is to find the most probable slot assignments $\hat{y}$ given utterance x:

$$\hat{y} = \operatorname*{argmax}_{y \in \mathcal{Y}(x)} p(y|x) \tag{2}$$

where $\mathcal{Y}(x)$ is the entire search space of slot assignments of x. Table 1 illustrates some example intent and slot types in *"Weather"* and *"Place"* domains. For example, if a user asks:

"*how is the weather in mountain view today*"

The system should identify that the intent is *"Check Weather"* in *"Weather"* domain. The location of interest is *"Mountain View"*, and the date is *"today"*. For another utterance:

"*where is mashiko in seattle*"

the user intent should be *"Find Place"* in *"Place"* domain. The location is *"Seattle"*, and the place name is *"Mashiko"*.

The approaches for slot filling range from generative models such as hidden Markov models [18, 19], discriminative classification methods [2, 20, 21, 22], probabilistic context free grammars [23, 24], unsupervised induction [25], and more recently deep learning methods [26, 27, 28].

3. DISTRIBUTED FRAMEWORK

Figure 1 depicts the overall architecture of our proposed framework. In this framework, each domain performs SLU independently. One can easily plug in new domains and service providers. Extensible domain-independent extractors are introduced to extract generic information (shared across multiple domains) from the user utterances, so that each domain can build their own SLU models rapidly.

3.1. Distributed Domain Selection

In the distributed framework, each domain has its own SLU component. When a user utterance is received, all competing domains process it in parallel without intervening others. Formally, for a given speech utterance x, the SLU component in each domain chooses to accept it or not:

$$\hat{c} = \operatorname*{argmax}_{c \in \{0,1\}} q(x, c) \tag{3}$$

where $c = 1$ means accept and 0 otherwise, and $q(x, c)$ is the confidence value of choosing c. If a probabilistic model is applied, $q(x, c) = p(c|x)$. Different from the centralized baseline, we use Bing queries as negative training examples for each domain in order to simulate the distributed scenario. This strategy differs from Eq.(1) in that each domain makes their decisions independently. Moreover, in the training phase, each domain is free to define its own training data, feature space, and classification algorithms. This enables all domains to concentrate on their particular interests, and reduces the complexity of modelling all domains at once.

Finally, when a new domain is introduced to the framework, it is not necessary to update training and decision phases of existing domains. Therefore extensibility can be enhanced at low cost.

In this paper we use Support Vector Machines (SVM) [29] implemented by SVM$^{\text{light}}$ toolkit [30] for binary classification of domain selection.

3.2. Domain Independent Extractors

Although the distributed framework can incorporate new domains and service providers flexibly, it is challenging and costly to build a reasonable SLU component for each domain from scratch. This issue obstacles our goal of rapidly adding new domains. We observe that there exists some common types of semantic annotations that are useful for multiple domains, such as locations, temporal expressions, and

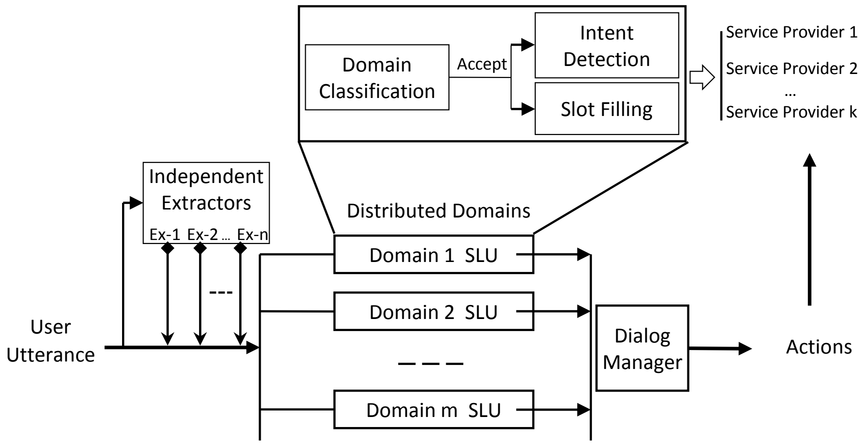


Fig. 1: Framework Overview. Ex$_i$ represents arbitrary independent extractors.

so on. For instance, for the two example utterances mentioned in Section 2, extracting locations is critical in both cases. It is expensive to duplicate a location extractor for each of them. To this end, we decouple generic information extraction and domain-specific SLU by introducing a new component, domain-independent extractors, into the framework. Given a speech utterance, each extractor can extract useful information with their particular expertise. For example, we can have one extractor focusing on extracting location entities, and another extractor that identifies and normalizes temporal expressions. Again, each extractor performs extraction in parallel. Therefore new extractors can be plugged in arbitrarily. Service providers can then build their own SLU parser rapidly on top of the generic extraction results.

3.3. Slot Filling Parser

In this paper, we cast the problem of slot filling to sequential labelling task with *B-I-O* schema. Each word is assigned with one of *B-X*, *I-X*, and *O* tags, where *X* means slot type, and *B,I* and *O* indicate the word is beginning of, within, and out-of a slot, respectively. The slots y for utterance x then becomes a sequence of labels $(y_1, y_2, ..., y_n)$, where n is the number of words in x. For instance, for the sentence "*how is the weather in mountain view today*", we can have the following *B-I-O* label sequence:

...	weather	in	mountain	view	today	y
...	*O*	*O*	*B-Location*	*I-Location*	*B-Date*	x

As an example implementation, we use linear-chain Conditional Random Fields (CRFs) [31] to model this labelling problem. In the case of first-order CRFs, the probability of assignment $\hat{y}$ for a given utterance x is:

$$p(\hat{y}|x) = \frac{1}{Z(x_i)} exp(\sum_i \sum_k (\theta_k \cdot f_k(y_{i-1}, y_i, x)) \quad (4)$$

where f_k is k-th feature function, θ_k is its weight, and Z(x) is a normalization factor. There are two types of features that can be used: unigram feature only depends on the current label y_i, whereas bigram feature dependes on both y_i and y_{i-1}. To estimate feature weights, we use stochastic gradient descent method to maximize the log-likelihood of training data.

4. EXPERIMENTS

In this section, we describe our experiments to test the feasibility and extensibility of the proposed framework, and demonstrate that by using domain-independent extractors, we can build high-performance slot fillers using only a small set of training instances.

4.1. Data Sets

The datasets used in our experiments are Microsoft Cortana-related corpus and the well-known ATIS corpus [32]. The Cortana-related data consists of 7 domains: *alarm, calendar, communication, note, places, reminder, and weather*. To provide negative training examples for each domain in the distributed scenario, we also collected a set of Bing search queries. Table 2 summarizes the statistics about the data sets.

4.2. Distributed Domain Detection

In this experiment, we compare the performance of distributed domain detection against traditional centralized domain classification. As an example implementation of the

Framework	Alarm	Calendar	Comm.	Note	Places	Reminder	Weather	Overall (Accuracy)
Centralized	96.8	93.8	96.3	93.1	96.1	95.1	98.6	96.0
Distributed	93.0	89.6	93.0	88.5	91.0	90.0	96.0	91.9
Distributed w/ Data from Competing Domains	96.7	94.2	96.3	93.5	96.9	95.1	98.8	96.3

Table 3: Overall performance (%) of *Centralized* and *Distributed* domain detection.

Data Set	Train	Test
Cortana-related Data	∼21,500 / domain	∼2,500 / domain
Bing Queries	100,000	-
ATIS	4,978	893

Table 2: Data sets statistics

Framework	Domain	Domain+Intent	Domain+Intent +Slot Filling
Centralized	96.0	91.1	84.9
Distributed	91.9	87.4	83.5

Table 4: End-to-end performance (%) of *Centralized* and *Distributed* framework.

centralized framework, we consider the task of domain classification as a multi-class classification problem, and train SVMs models with one-vs.-rest strategy on the Cortana-related data. In the distributed scenario, for each domain, we train a binary classifier in isolation using SVMs to predict whether a utterance belongs to it. Both methods employ tri-gram features and linear kernel.

The overall results on Cortana test set are summarized in Table 3. As we can see, the distributed method achieves slightly lower performance than the centralized one, although its negative training examples are random Bing search queries, which are noisy for differentiating positive examples for each domain. In addition, when we add the training data from other domains as negative training examples for each binary classifier, the performance becomes marginally better than the centralized one (row 4 in Table 3). It supports our assumption that distributed domain selection can perform as well as centralized one for multiple domains.

To further simulate the scenario where an emerging domain needs to be added, we collected a set of Cortana queries that express greetings to the agent, such as "*hello, Cortana*". We then consider those queries as a new domain (*Greeting*), train a binary classifier for the new domain in the same way as the existing domains, and plug it into the distributed system. The *Greeting* model correctly identified 3,193 instances out of 3,200, and the overall accuracy of existing domains is the same as before. This result demonstrates that the distributed method is capable of incorporating new domains without intervening in existing ones.

Finally, we compare the end-to-end performance of centralized and distributed frameworks by adding intent classification and slot filling components. We trained intent classifiers for each domain using SVMs model with tri-gram features and linear kernel, and slot filling parsers using CRFs with words in the window of size 2 as features. The results are summarized in Table 4. The scores of *Domain* and *Domain+Intent* are measured by accuracy, and the final end-to-end performance is measured by F_1 metric, where system-output slots are ignored if either domain or intent is incorrect.

feature type	words features	extractor features
unigram	w_{-2}, w_{-1}, w_0, w_1, and w_2	e_{-2}, e_{-1}, e_0, e_1, and e_2
bigram	true	

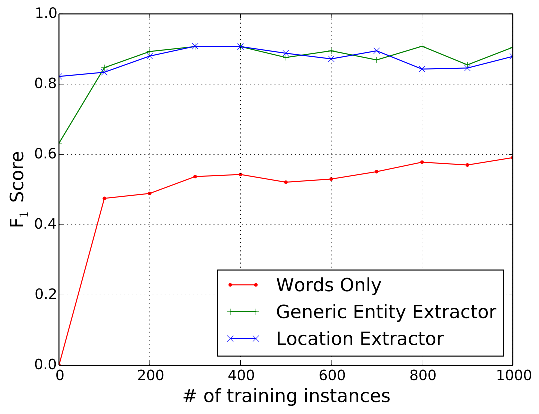
Table 5: Slot filling feature template. w_0 represents the current word, and e_0 denotes the tag of the current word provided by independent extractor.

4.3. Slot Filling Based on Generic Extractors

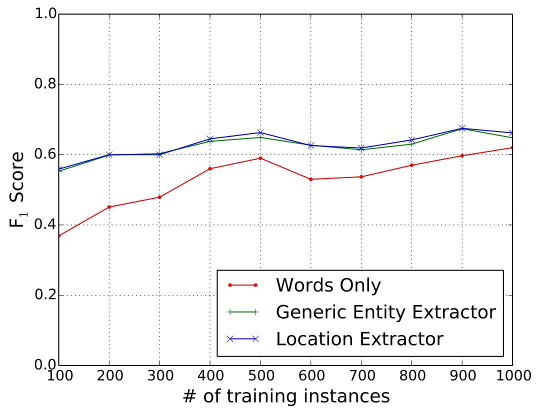
In this experiment, we test the advantage of using independent extractors. We consider two test cases: location slots in *Weather* domain of Cortana-related data, and all slot types in ATIS corpus. Three independent extractors are employed:

- Ex_1: a CRFs-based location extractor trained from an earlier Cortana-related corpus including *hotels*, *restaurant* and *movies* domains [2].
- Ex_2: a generic entity extractor trained using structured perceptron [33] over more than a million hand-annotated labels in 25 entity types such as location, organiation, etc.
- Ex_3: a named entity gazetteer for ATIS corpus.

Based on those extractors we train slot filling parsers using first-order linear-chain CRFs, and compare the performance of using only words features and using the two extractors respectively. We use Ex_1 and Ex_2 for the *Weather* domain data set and collapsed all location-related entity types from Ex_2 such as city and state to *location*, and use Ex_2 and Ex_3 for ATIS corpus. Table 5 summarizes the feature template that we used for both test cases. The learning curves with training size varying from 100 to 1,000 are plotted in Figure 2 and Figure 3. When the training size equals to 0 in Figure 2a, we evaluate the output of Ex_1 and Ex_2 against the ground-truth of *Weather* data directly. For both corpora, we can clearly see



(a) *Location* slot in *Weather* domain.



(b) All slots in *Weather* domain.

Fig. 2: Learning curves of slot filling on Weather domain using independent extractors.

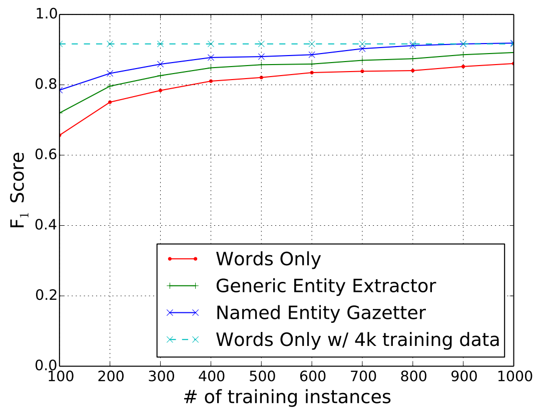


Fig. 3: Learning curves of slot filling on ATIS corpus using independent extractors.

that using the extractors the performance of slot filling can be boosted substantially when only a small amount of training data is used.

5. CONCLUSIONS AND FUTURE WORK

We presented a distributed conversational understanding framework with domain-independent extractors. This framework enables each domain develop and apply its SLU model independently. Therefore one can easily incorporate new domains without modifying existing ones. Furthermore, we introduce a new component, domain-independent extractors, to extract useful information that can be shared by all SLU developers. With these extractors, each new domain or service provider can develop high-performance domain-specific SLU models efficiently. Our preliminary experiments demonstrated the advantages of the proposed framework compared to conventional framework.

For future work, we plan to incorporate more domains and service providers, and develop more independent extractors, such as cuisine type extractor that can be useful for restaurant-related service providers. Furthermore, we will expand this framework for multiple turns of user-machine conversation.

6. REFERENCES

[1] "Windows phone store," `http://en.wikipedia.org/wiki/Windows\_Phone\_Store`.

[2] X. Li, Y.-Y. Wang, and G. Tur, "Multi-task learning for spoken language understanding with shared slots," in *INTERSPEECH*, 2011, pp. 701–704.

[3] G. Tur, R. E. Schapire, and D. Hakkani-Tür, "Active learning for spoken language understanding," in *Proceedings of the ICASSP*, Hong Kong, May 2003.

[4] G. Tur, "Model adaptation for spoken language understanding," in *Proceedings of the ICASSP*, Philadelphia, PA, May 2005.

[5] A. Celikyilmaz, D. Hakkani-Tür, and G. Tur, "Multi-domain spoken language understanding with approximate inference," in *Proceedings of the Interspeech*, 2011.

[6] D. Hakkani-Tür, G. Tur, L. Heck, A. Fidler, and A. Çelikyilmaz, "A discriminative classification-based approach to information state updates for a multi-domain dialog system," in *INTERSPEECH*, 2012.

[7] B.-S. Lin, H.-M. Wang, and L.-S. Lee, "A distributed architecture for cooperative spoken dialogue agents with coherent dialogue state and history," in *in Proc. ASRU-99*, 1999.

[8] B.-S. Lin, H.-M. Wang, and L.-S. Lee, "A distributed agent architecture for intelligent multi-domain spoken dialogue systems," p. 12171230, 2001.

[9] M. Nakano, S. Sato, K. Komatani, K. Matsuyama, K. Funakoshi, and H. G. Okuno, "A two-stage domain selection framework for extensible multi-domain spoken dialogue systems," in *SIGDIAL Conference*, 2011, pp. 18–29.

[10] K. Komatani, N. Kanda, M. Nakano, K. Nakadai, H. Tsujino, T. Ogata, and H. G. Okuno, "Multi-domain spoken dialogue system with extensibility and robustness against speech recognition errors," in *Proceedings of the 7th SIGdial Workshop on Discourse and Dialogue*, 2006, pp. 9–17.

[11] G. D. Fabbrizio, G. Tur, D. Hakkani-Tür, M. Gilbert, B. Renger, D. Gibbon, Z. Liu, and B. Shahraray, "Bootstrapping spoken dialogue systems by exploiting reusable libraries," in *Natural Language Engineering*, 2008, pp. 313–335.

[12] R. E. Schapire and Y. Singer, "Boostexter: A boosting-based system for text categorization," *Machine Learning*, vol. 39, no. 2/3, pp. 135–168, 2000.

[13] N. Gupta, G. Tur, D. Hakkani-Tür, S. Bangalore, G. Riccardi, and M. Rahim, "The AT&T spoken language understanding system," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 1, pp. 213–222, 2006.

[14] I. Zitouni, H.-K. J. Kuo, and C.-H. Lee, "Boosting and combincation of classifiers for natural language call routing systems," *Speech Communication*, vol. 41, no. 4, pp. 647–661, 2003.

[15] P. Haffner, G. Tur, and J. Wright, "Optimizing SVMs for complex call classification," in *Proceedings of the ICASSP*, Hong Kong, April 2003.

[16] R. Sarikaya, G. E. Hinton, and B. Ramabhadran, "Deep belief nets for natural language call-routing," in *Proceedings of the ICASSP*, Prague, Czech Republic, 2011.

[17] Y. Dauphin, G. Tur, D. Hakkani-Tür, and L. Heck, "Zero-shot learning and clustering for semantic utterance classification," in *Proceedings of the ICLR*, 2014.

[18] R. Pieraccini, E. Tzoukermann, Z. Gorelov, J.-L. Gauvain, E. Levin, C.-H. Lee, and J. G. Wilpon, "A speech understanding system based on statistical representation of semantics," in *Proceedings of the ICASSP*, San Francisco, CA, March 1992.

[19] Y. He and S. Young, "A data-driven spoken language understanding system," in *Proceedings of the IEEE ASRU Workshop*, U.S. Virgin Islands, December 2003, pp. 583–588.

[20] R. Kuhn and R. De Mori, "The application of semantic classification trees to natural language understanding," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, pp. 449–460, 1995.

[21] Y.-Y. Wang and A. Acero, "Discriminative models for spoken language understanding," in *Proceedings of the ICSLP*, Pittsburgh, PA, September 2006.

[22] G. Tur, D. Hakkani-Tür, and L. Heck, "What is left to be understood in ATIS?," in *Proceedings of the IEEE SLT Workshop*, Berkeley, CA, 2010.

[23] S. Seneff, "TINA: A natural language system for spoken language applications," *Computational Linguistics*, vol. 18, no. 1, pp. 61–86, 1992.

[24] W. Ward and S. Issar, "Recent improvements in the CMU spoken language understanding system," in *Proceedings of the ARPA HLT Workshop*, March 1994, pp. 213–216.

[25] Y.-N. Chen, W. Wang, and A. Rudnicky, "Unsupervised induction and filling of semantic slots for spoken dialogue systems using frame-semantic parsing," in *Proceedings of the IEEE ASRU Workshop*, 2013, pp. 120–125.

[26] L. Deng, G. Tur, X. He, and D. Hakkani-Tür, "Use of kernel deep convex networks and end-to-end learning for spoken language understanding," in *In Prooceedings of the IEEE SLT Workshop*, Miami, FL, December 2012.

[27] K. Yao, B. Peng, G. Zweig, D. Yu, X. Li, and F. Gao, "Recurrent conditional random field for language understanding," in *Proceedings of the IEEE ICASSP*, 2014.

[28] P. Xu and R. Sarikaya, "Convolutional neural network based triangular crf for joint intent detection and slot filling," in *Proceedings of the IEEE ASRU*, 2013.

[29] V. N. Vapnik, *Statistical Learning Theory*, John Wiley and Sons, New York, NY, 1998.

[30] "Svmlight support vector machine toolkit," http://svmlight.joachims.org.

[31] J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proceedings of the ICML*, Williamstown, MA, 2001.

[32] P. J. Price, "Evaluation of spoken language systems: The ATIS domain," in *Proceedings of the DARPA Workshop on Speech and Natural Language*, Hidden Valley, PA, June 1990.

[33] M. Collins, "Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms," in *Proceedings of EMNLP*, 2002.