EASY CONTEXTUAL INTENT PREDICTION AND SLOT DETECTION

A. Bhargava[1]*, *A. Celikyilmaz*[2], *D. Hakkani-Tür*[3], *and R. Sarikaya*[2]

[1]University of Toronto, [2]Microsoft, [3]Microsoft Research

ABSTRACT

Spoken language understanding (SLU) is one of the main tasks of a dialog system, aiming to identify semantic components in user utterances. In this paper, we investigate the incorporation of context into the SLU tasks of intent prediction and slot detection. Using a corpus that contains session-level information, including the start and end of a session and the sequence of utterances within it, we experiment with the incorporation of information from previous intra-session utterances into the SLU tasks on a given utterance. For slot detection, we find that including features indicating the slots appearing in the previous utterances gives no significant increase in performance. In contrast, for intent prediction we find that a similar approach that incorporates the intent of the previous utterance as a feature yields relative error rate reductions of 6.7% on transcribed data and 8.7% on automatically-recognized data. We also find similar gains when treating intent prediction of utterance sequences as a sequential tagging problem via SVM-HMMs.

Index Terms— spoken language understanding, slot detection, intent prediction, contextual models

1. INTRODUCTION

Spoken language understanding (SLU) is an important part of modeling human-computer dialogs, aiming at determining a user's intents from their utterances, known as intent prediction, as well as identifying relevant actionable pieces of the utterance, known as slot detection [1]. For example, given the utterance "Show me Joss Whedon's latest movies", the user's intent is to see a list of content matching the query, and "Joss Whedon" fills one slot (director) and "movies" fills another (media type). Once the intents and slots are determined, they are passed to the dialog manager which forms a query to the back-end and determines the response given to the user.

Traditionally, both of these tasks are considered one utterance at a time by the SLU process (due in part to the fact that gathering the session data requires a live, deployed system, which is not available when building the initial SLU models). Both intent prediction and slot detection systems are given isolated utterances and asked to label them with the appropriate information as the task requires. This runs counter to the common means by which these utterances are gathered: by initiating *sessions* with users and recording their speech (we can see these sessions as crude human-computer conversations). Ultimately, each utterance occurs within the *context* of a larger discourse between a human and an automated agent. Fig. 1 exemplifies this, showing a case where the user makes clear references to previous utterances or the system's responses to the same.

In this paper, we examine the effect of incorporating information from previous intra-session utterances, to which we refer broadly as context, into both intent prediction and slot detection. Context is an important viable source for new information, which is especially important in the case of noisy automatically-recognized data. For example, the true (transcribed) utterance in Table 1 contains the word "like", which would strongly indicate the "find similar content" intent, but this word is lost during the automatic recognition process. This inherent noise in the ASR process makes any alternative hints extremely valuable; in such situations the context may provide additional information that may help find the correct intent. Traditionally, it has been the dialog manager's role to handle the utterance in context, but incorporating this information earlier (during SLU) can help avoid cascaded errors further down the pipeline.

Source	Utterance
Transcribed	`tv shows like buffy the vampire slayer`
Recognized	`tv show with play buffy the vampire slayer`

Table 1. An example of noise due to ASR. In this case, it is difficult to correctly determine the correct intent ("find similar content") from the recognized utterance alone.

We investigate two different approaches for integrating context into SLU for intent and slot determination: (*i*) modeling sessions as a sequence for classification; and (*ii*) feature engineering. We perform several experiments on both transcribed and automatically-recognized utterances. Using conditional random fields for slot detection, we find that incorporating features based on the appearance of slots in previous utterances yields no significant benefit. In contrast, we are able to increase the performance of intent prediction based on both classification and sequential tagging systems. We demonstrate higher reductions in error rate over automatically-recognized data, and find that the classification and sequential tagging approaches perform equally in this regard.

2. BACKGROUND

2.1. Intents and slots

Intents are global properties of utterances. Intents signify the goal of the user and vary across domains, but ultimately, a dialog system must at some point make a determination as to what a user wants given an utterance. This is of particular relevance to command-and-control systems, where each utterance signifies a command upon which the system is to act. To provide an analogy grounded in software development, intents map roughly to functions that are meant to be called: if the intent is detected to be "find content", call the relevant `find_content` function.

Intent prediction has traditionally been modeled as an utterance classification task, mainly using local information about the utterance. Early work with discriminative classification algorithms was conducted on the AT&T HMIHY system, using for example boosting [2]

*The first author performed the work during an internship at Microsoft

U	Intent	Transcribed	Recognized
u_1	get clip	show me the [**firefly**]$_{content-name}$ [**trailer**]$_{type}$	show me the [**firefly**]$_{content-name}$ [**trailer**]$_{type}$
u_2	find info	who directed [**it**]$_{content-name-ref}$	who directed [**it**]$_{content-name-ref}$
u_3	find content	what else has [**he**]$_{director-ref}$ done	what else has [**he**]$_{director-ref}$ done
u_4	play content	play [**the avengers**]$_{content-name}$	plane [**avatars**]$_{content-name}$

Fig. 1. An example session, including both transcribed and recognized versions of the utterances $u_1, \ldots, u_4$. Each utterance has an associated intent, while the slots are shown within each utterance. The slots are annotated on the transcribed data and transferred automatically to the ASR version.

or max-margin classifiers [3]. Cox [4] proposed the use of generalized probabilistic descent (GPD), corrective training (CT), and linear discriminant analysis (LDA).

Slots, on the other hand, exist within utterances. Slots are local properties in the sense that they span individual words rather than whole utterances. In the context of human-computer dialogs, and in particular command and control, the words that fill slots tend to be the only semantically loaded words in the utterance (i.e., the other words are function words). Slots represent actionable content; continuing the software development analogy, slot values map to values passed as function parameters: we would pass the value "Joss Whedon" to the "director" parameter, for example as `find_content(director='Joss Whedon')`.

Slot detection in SLU has commonly been approached using a classification method for filling frame slots given an application domain. These approaches include, among others, generative models such as hidden Markov models [5], discriminative classification methods [6, 7, 8], and probabilistic context-free grammars [9, 10].

2.2. Session modeling

In the dialog manager, belief-state update is a common method for modeling context. One of the earliest statistical approaches to multi-turn interpretation was the statistical discourse model by Miller et al. [11] that introduced a mapping from the pre-discourse meaning and the previous meaning (as determined after the previous user turn) to the post-discourse meaning. Later on, Levin and Pieraccini [12] proposed using Markov decision process (MDP) as a dialog model. However, MDPs assume that dialog states are observable, so they do not account for any uncertainty in the dialog history or the user state. The application of partially-observable Markov decision processes (POMDP) was suggested for dialog modeling [13] to handle uncertainty. Thomson [14] used dynamic Bayesian networks and separated the belief estimation based on observations till the current time from the estimation of the optimum action.

In this paper, similar to dialog modeling, we focus on leveraging context, but at the level of SLU modeling. Considering context at the SLU level is important for several reasons:

- As the SLU process occurs earlier in the dialog system's architecture, errors during SLU can be cascaded throughout the rest of the system. Any improvements to the SLU will help minimize overall system error.
- Not all conversational applications require dialog modeling; in the traditional architecture, such cases would miss out on contextual information.
- Similarly, other downstream applications can benefit from improved SLU performance
- Where other dialog system components can be very tied to specific application scenarios and subject to revision based on changes in knowledge sources or user interface, general contextual SLU can be insulated from such sensitivities.

3. METHOD

3.1. Session data

Our session data are internally collected from real-use scenarios of a spoken dialog system. We focus here on the domain of audiovisual media, including movies and television shows. The user is expected to interact by voice with a system that can perform a variety of tasks in relation to such media, including (among others) browsing, searching, querying information, and playing.

In total, our corpus has a total of 27565 utterances split into 6390 sessions. There are 28 possible intents and 26 possible slot types. The most frequent intents include *find content*, *play content*, *find similar*, *filter*, and *start over*; top slot types include *content name*, *content type*, *genre*, *stars*, and *content description*.

Our corpus consists of a series of utterances, where each utterance belongs to a particular session. Each utterance is annotated with the intent as well as slots occurring in the utterance. Our corpus contains both transcribed (TRA) and speech-recognized (ASR) versions of the utterance; since the intent is a global property of the utterance, it applies trivially to both versions. We transfer the correct slots from TRA to ASR using the word-by-word alignment of hypothesized words to the reference transcriptions. Once the words are aligned, labels are transferred from the reference to the hypothesized words. Since some tokens may be lost in this step (for example, a slot-initial word deleted in the ASR output), we normalize annotations on the ASR side in order to obtain a well-formed set. Fig. 1 shows an example session consisting of four utterances with both TRA and ASR versions along with the intents and slots.

3.2. Learning approaches

The work in this paper employs three machine learning methods:

Support vector machines (SVMs) [15], in their most basic formulation, are a binary classification method based on the intuition of maximizing the margin around the classification boundary. SVMs have seen extensive use for language tasks, owing to their simplicity of use, ability to incorporate many features, and strong performance. SVMs can also be extended for multi-class scenarios and can be used with non-linear kernels. We use linear kernels as provided in the very fast LIBLINEAR [16] package, which allows for multi-classification.

SVMs can also be applied to structural prediction. In particular, we consider their application to the training of **hidden Markov mod-**

els (SVM-HMMs) [17]. In contrast to standard (generative) HMMs, this method allows for the incorporation of numerous (potentially long-range) features. SVM-HMMs solve the following optimization problem:

$$\begin{aligned} \min_{\mathbf{w},\xi} \quad & \tfrac{1}{2}\left|\mathbf{w}\right|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{s.t.} \quad & \\ \forall i \forall \mathbf{y}: \quad & \sum_{j=1}^{l}\Phi\left(x_j^i, y_{j-1}^i, y_j^i\right) \geq \\ & \sum_{j=1}^{l}\Phi\left(x_j^i, y_{j-1}, y_j\right) + \Delta\left(\mathbf{y}^i, \mathbf{y}\right) - \xi_i \end{aligned}$$

where $\mathbf{w}$ is the weight vector to be learned, $\mathbf{x}^i = (x_1^i, \ldots, x_j^i)$ is observation sequence i, $\mathbf{y}^i = (y_1^i, \ldots, y_j^i)$ is the correct tag (hidden state) sequence corresponding to $\mathbf{x}^i$ and ξ_i are the slack variables for each input data point that measure the degree of misclassification of the data x_i and help if the problem is not linearly separable. The candidate tag sequences $\mathbf{y} = (y_1, \ldots, y_j)$ are generated using the Viterbi algorithm. Φ is a feature function that returns a vector consisting of two components: a feature vector constructed from the observation state (as (1) a feature vector constructed from the observation state as specified by the user; and (2) a vector of binary transition features, having one feature for each possible hidden-state transition with all of them set to 0 except for the one corresponding to the transition (y_{j-1}^i, y_j^i) given to Φ. Δ is a loss function equivalent to the number of incorrect tags in the candidate sequence $\mathbf{y}$. As with regular SVMs, SVM-HMMs have been shown to work well on language tasks [18, 19, 20, 21]. We use the SVM^{hmm} package[1].

Lastly, **conditional random fields (CRFs)** [22] are discriminative (conditionally-trained) graphical models that have been used effectively for sequence labeling tasks, a standard example of which is part-of-speech tagging. CRFs also allow the incorporation of many more features than is possible with HMMs. Using similar notation as with SVM-HMMs above, CRFs define the conditional probability of $\mathbf{y}$ as:

$$p_{\mathbf{w}}(\mathbf{y}|\mathbf{x}) = \tfrac{1}{Z_{\mathbf{w}}}\prod_{t=1}^{T}\phi^{(t)}\left(y_t, y_{t-1}, \mathbf{x}\right)$$

where $\phi_t(\cdot)$ is the local potential function represented with maximum cliques of the graph. The partition function $Z_{\mathbf{w}}(\mathbf{x}) \triangleq \sum_{\mathbf{y}}\prod_{t=1}^{T}\phi_t(y_t, y_{t-1}, \mathbf{x})$ ensures that the probability of all state sequences sum to one. The ϕ_t functions factorize based on feature functions f_k:

$$\begin{aligned} \phi^{(t)}\left(y_t, y_{t-1}, \mathbf{x}\right) &= \overbrace{\phi_{\mathbf{1}}^{(t)}\left(y_t, \mathbf{x}\right)}^{observation} \cdot \overbrace{\phi_{\mathbf{2}}^{(t)}\left(y_t, y_{t-1}\right)}^{transition} \\ \phi_{\mathbf{1}}^{(t)}\left(y_t, \mathbf{x}\right) &= \exp\left(\textstyle\sum_k \mathbf{w}_1^k f_1^k(y_t, \mathbf{x}_t)\right) \\ \phi_{\mathbf{2}}^{(t)}\left(y_t, y_{t-1}\right) &= \exp\left(\textstyle\sum_k \mathbf{w}_2^k f_2^k(y_t, y_{t-1})\right) \end{aligned}$$

where $\mathbf{w}$ is the weight vector and $f_1^k(y_t, \mathbf{x}_t)$ and $f_2^k(y_t, y_{t-1})$ are feature functions that encode topics (slots) of the current word and state transitions in sequence at the current index t. We use the CRFSGD[2] package [23] for our CRF implementation.

3.3. Incorporating contextual information

For both intent prediction and slot detection, we aim to incorporate information from the previous utterance. In particular, for each task, we aim to incorporate the values of that task on the previous utterance in the session.

[1] http://www.cs.cornell.edu/people/tj/svm_light/svm_hmm.html

[2] http://leon.bottou.org/projects/sgd

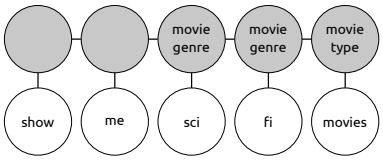

Fig. 2. Slot detection as a sequential tagging problem with CRFs. Hidden states are shown shaded.

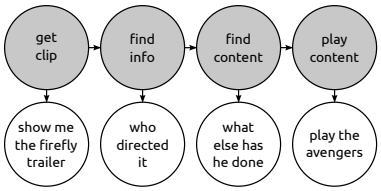

Fig. 3. Contextual intent prediction as a sequential tagging problem with HMMs. Hidden states are shown shaded.

For slot detection, which we treat as a sequential tagging problem as shown in Fig. 2, we apply CRFs. The baseline implementation uses lexical features consisting of unigrams in a five-word window around the current word. To incorporate contextual information, we add binary features for all possible slot types that might have occurred in the previous utterance. The features corresponding to slots occurring in the previous utterance are set to 1 while all others are set to 0.

For intent prediction, we try two separate approaches. First, we treat it as an utterance classification problem, with our baseline using bag-of-n-gram features to train an SVM. In this case, we can add context by adding additional features to indicate the intent of the previous utterance. We have one new feature for each possible intent, with the one corresponding to the intent of the previous utterance set to 1 and all others set to 0. We also include a feature that indicates that the given utterance is the first in the session, which handles the case that there is no previous utterance.

Second, we can treat intent prediction as a sequential tagging problem, as shown in Fig. 3. Each utterance is a candidate to be tagged, with the utterances occurring in sequence as defined by the session. We use SVM-HMMs in this case, as they allow for easier processing of sparse feature vectors.

4. EXPERIMENTS

4.1. Setup

The data corpus described above in Section 3.1 is split into training, development, and test sets comprising 80%, 10%, and 10% of the full corpus respectively. The test set is held out and used only for evaluation. We use the development set to tune hyperparameters, and then merge it with the training set to train the final model used for testing. We report the results on the held-out test set here.

For the slot detection (CRF) and intent prediction (SVM classification) cases, we aim to incorporate information from the previous utterance into the task on the current utterance. We aim to gauge both the overall potential of incorporating information from previous utterances as well as testing this idea in a real-world scenario. Considering first intent prediction, this leads us to test two alternative cases: first we test the inclusion of the previous utterance's intention with the intent drawn directly from the corpus; this roughly represents an upper bound on how much improvement we can hope to achieve by

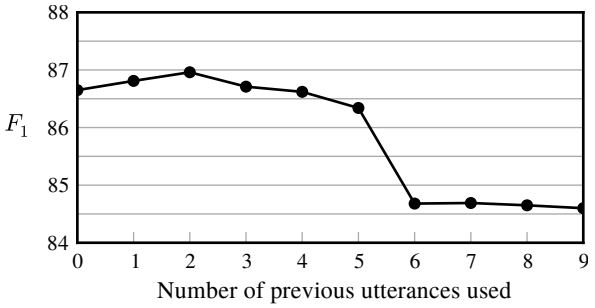

Fig. 4. Slot detection F_1-score with context size, showing a statistically insignificant increase in performance for immediately previous utterances and decreases in performance for larger context windows.

incorporating contextual information in this way.

Second, we test the more realistic case where the previous intent is instead generated by an intent prediction model. This second case requires that first a model is trained without any contextual features; this model is then used to generate hypothesized intents that can be used to train a new model. Note that this requires care to be taken in terms of experimental architecture: the first, non-contextual model, when trained on the training set, will obviously produce excellent results when evaluated on the training set (which is required to produce the hypothesized intents for training the subsequent contextual model). To produce realistic results, we generate the hypothesized intents by splitting the training set into 10 folds. We then successively hold out one fold and train a (non-contextual) model on the remaining folds, and then evaluate on the held-out fold. The results are all merged together to produce hypothesized intents for training the subsequent contextual model.

The same process is repeated with slot detection, but of course with slots and slot detection models rather than intents and intent prediction models. Note that this methodology emulates a real-world two-stage model in which a first stage predicts (comparatively) naive outputs, while the second stage incorporates the outputs for previous utterances from the first stage.

Note that the SVM-HMM method for intent prediction does not require the above considerations. In effect, it is most comparable to the SVM experiments that use the predicted previous intent, as it does not use the actual previous intent during test time at all, only its own predicted intents.

4.2. Slot detection

Fig. 4 shows the results for slot detection. Here, we consider the effect of including slot types appearing in the previous n intra-session utterances. We evaluate the slot detection performance using the F_1-score, which is defined in terms of precision P and recall R as $F_1 = \frac{2PR}{P+R}$. Ultimately, while we see a small increase looking at the previous two utterances, these are insignificant, and performance decreases if we look further back than that.

4.3. Intent prediction

Table 2 shows the results for intent prediction, evaluated using accuracy over all utterances in the test set, of the SVM with either the lexical features only or the lexical features combined with the previous intent features (for both oracle and actual previous intents). As expected, we see a drop in accuracy on the ASR data compared to the TRA data due to the additional noise in and hypothetical nature of the ASR outputs. As for the effect of incorporating the previous intent, we see a 6.7% error rate reduction over the TRA baseline and an 8.7% error rate reduction over the ASR baseline. Combining these results with the fact that the predicted-intent accuracies are very close to the oracle-intent accuracies, we can see that this approach of incorporating the previous intent is indeed effective.

	TRA	ASR
LEX	97.1	93.1
ORCLPREV	97.3	93.9
PREDPREV	97.3	93.7

Table 2. Utterance accuracies in percentages for intent prediction. LEX indicates lexical features only; ORCLPREV indicates lexical features plus the oracle (actual) intent of the previous utterance; and PREDPREV indicates lexical features plus the predicted intent of the previous utterance. TRA indicates the corpus of transcribed utterances while ASR indicates the corpus of ASR outputs.

Lastly, we compare the simple additional-feature approach to the more complex SVM-HMM modeling method. We find no statistical significance between the SVM-HMM and the SVM with predicted previous intents; so while the SVM-HMM provides an improvement over using lexical features only, SVMs are a better choice due to their lower modeling complexity and commensurate speed.

Furthermore, it's important to note that the SVM-HMM result is an unrealistic scenario, although for different reasons than the oracle-intent scenario for the vanilla SVM. The SVM-HMM approach uses the Viterbi algorithm, which looks through the whole session (including "future" utterances) to determine the best overall intent sequence. Obviously, during real-world use, we do not have access to the full session, so even this result is an upper bound for the SVM-HMM's performance for this task.

Lastly, we note that the success of our approach to contextualizing intent prediction makes sense in light of the effect of the previous intent on the distribution of current intents. For example, 87% of first utterances in a session are annotated as having the "find content" intent, but this drops to 49% if the previous utterance was also a "find content" utterance. Similarly, if the previous utterance was a "play content" utterance, the most likely (32%) intent for the current utterance is also "play content". This very clearly demonstrates the contextual sensitivity of the intent prediction task.

5. CONCLUSION

In this paper, we examined the effect of adding session context to the SLU tasks of intent prediction and slot detection. Our simple feature engineering–based approach involved adding features that indicated the state (whether slot types or intent) of previous utterances; while this approach yielded no significant improvement for CRF-based slot detection, we found significant error rate reductions of 6.7% and 8.7% for SVM-based intent prediction on transcribed and automatically-recognized data, respectively. We found a similar increase when intent prediction was modeled as a sequential tagging problem using SVM-HMMs. These results indicate that incorporating context into the SLU process rather than leaving it for the dialog manager is a viable and effective option. Possible feature work includes experiments using joint modeling of intent and slots to explore much higher-order models.

6. REFERENCES

[1] G. Tur and R. De Mori, Eds., *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*, John Wiley and Sons, New York, NY, USA, 2011.

[2] A. L. Gorin, G. Riccardi, and J. H. Wright, "how may i help you?," *Speech Communication*, vol. 23, pp. 113–127, 1997.

[3] V.N. Vapnik, "Statistical learning theory," *John Wiley and Sons, New York, NY*, 1998.

[4] S. Cox, "Discriminative techniques in call routing," in *Proceedings of ICASSP, Hong Kong*, April 2003.

[5] 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, USA, March 1992.

[6] 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.

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

[8] C. Raymond and G. Riccardi, "Generative and discriminative algorithms for spoken language understanding," in *Proceedings of the Interspeech*, Antwerp, Belgium, 2007.

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

[10] 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.

[11] S. Miller, D. Stallard, R. Bobrow, , and R. Schwartz, "A fully statistical approach to natural language interfaces," in *Proceedings of the ACL*, 1996.

[12] E. Levin and R. Pieraccini, "A stochastic model of computer-human interaction for learning dialogue strategies," in *Proceedings of Eurospeech*, 1997.

[13] J. Williams and S.J. Young, "Scaling up POMDPs for dialog management: The "summary POMDP" method," in *Proceedings of ASRU Workshop*, 2005.

[14] B. Thomson, *Statistical methods for spoken dialogue management*, Ph.D. thesis, University of Cambridge, 2009.

[15] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, pp. 273–297, September 1995.

[16] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.

[17] Y. Altun, I. Tsochantaridis, and T. Hofmann, "Hidden Markov support vector machines," in *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)*, Washington DC, USA, August 2003, Morgan Kaufmann Publishers Inc.

[18] S. Bartlett, G. Kondrak, and C. Cherry, "Automatic syllabification with structured SVMs for letter-to-phoneme conversion," in *Proceedings of ACL-08: HLT*, Columbus, Ohio, June 2008, pp. 568–576, Association for Computational Linguistics.

[19] B. Chang and D. Han, "Enhancing domain portability of chinese segmentation model using chi-square statistics and bootstrapping," in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, Cambridge, MA, October 2010, pp. 789–798, Association for Computational Linguistics.

[20] S. Kiritchenko and C. Cherry, "Lexically-triggered hidden markov models for clinical document coding," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Portland, Oregon, USA, June 2011, pp. 742–751, Association for Computational Linguistics.

[21] M. Verbeke, V. Van Asch, R. Morante, P. Frasconi, W. Daelemans, and L. De Raedt, "A statistical relational learning approach to identifying evidence based medicine categories," in *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, Jeju Island, Korea, July 2012, pp. 579–589, Association for Computational Linguistics.

[22] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proceedings of the Eighteenth International Conference on Machine Learning (ICML-2011)*, San Francisco, CA, USA, 2001, pp. 282–289, Morgan Kaufmann Publishers Inc.

[23] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of the 19th International Conference on Computational Statistics (COMPSTAT'2010)*, Paris, France, August 2010, pp. 177–187, Springer.