Discovering Dialogue Slots with Weak Supervision

Vojtěch Hudeček,[1] **Ondřej Dušek**[1] **and Zhou Yu**[2]
[1]Charles University, Faculty of Mathematics and Physics,
Institute of Formal and Applied Linguistics
[2] Columbia University, Department of Computer Science
{hudecek,odusek}@ufal.mff.cuni.cz, zy2461@columbia.edu

Abstract

Task-oriented dialogue systems typically require manual annotation of dialogue slots in training data, which is costly to obtain. We propose a method that eliminates this requirement: We use weak supervision from existing linguistic annotation models to identify potential slot candidates, then automatically identify domain-relevant slots by using clustering algorithms. Furthermore, we use the resulting slot annotation to train a neural-network-based tagger that is able to perform slot tagging with no human intervention. This tagger is trained solely on the outputs of our method and thus does not rely on any labeled data.

Our model demonstrates state-of-the-art performance in slot tagging without labeled training data on four different dialogue domains. Moreover, we find that slot annotations discovered by our model significantly improve the performance of an end-to-end dialogue response generation model, compared to using no slot annotation at all.

1 Introduction

Task-oriented dialogue systems typically use annotation based on *slots* to represent the meaning of user utterances (Young et al., 2013). Slots are attributes relevant to completing the task (e.g., *price*, *food type*, *area*). The sets of slots and their values typically need to be designed in advance by domain experts. Slots and their values are tracked over the course of the dialogue, forming dialogue state, which allows a dialogue system to plan the next actions effectively (Williams et al., 2013).

Getting raw data for dialogue system training is not difficult, especially if we restrict the target domain. A requirement for dialogue state labels makes this process much more costly. However, both traditional pipeline systems (Young et al., 2013) and end-to-end task-oriented architectures (Wen et al., 2017) typically require such annotation. While some systems use implicit, latent state representation and do not require annotation (Serban et al., 2016), the behavior of such systems is hard to interpret or control. There are several works aiming at keeping interpretability and reducing the annotation needs by automating it (Chen et al., 2014, 2015) or transferring annotation across domains (Zhao and Eskenazi, 2018; Coope et al., 2020), but they still require significant manual effort.

In this paper, we present a novel approach to discovering a set of domain-relevant dialogue slots and their values given a set of dialogues in the target domain (such as transcripts from a call center). Our approach requires no manual annotation at all in order to tag slots in dialogue data. This substantially simplifies dialogue system design and training process, as the developer no longer needs to design a set of slots and annotate their occurrences in training data. We discover slots by using unsupervised clustering on top of annotation obtained by domain-independent generic models such as a semantic frame parser or a named entity recognizer (NER). To illustrate our approach, let us consider an example given in Figure 1.

Figure 1: An utterance from the restaurant recommendation domain tagged with off-the-shelf frame semantic parser. Some tags are domain-relevant (shown in blue), but some are not (shown in gray).

Although the annotation is descriptive, it contains concepts irrelevant for the domain under consideration. Our method selects only relevant slot candidates (depicted in blue). Slots discovered by our approach can then be used to design or adapt the database backend for the target domain.

Our contributions can be summarized as follows:

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 2430–2442
August 1–6, 2021.

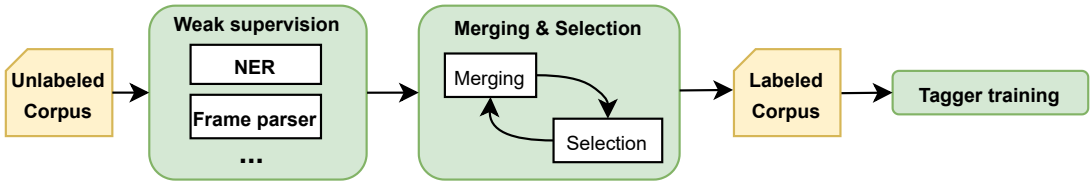

Figure 2: Illustration of our pipeline. First, we analyze an unlabeled in-domain corpus with supplied domain-agnostic linguistic annotation models, such as a frame-semantic parser or NER (Section 3.1). This results in slot candidates. Next, we iteratively merge and select slot candidates to obtain domain-relevant slots (Sections 3.2.2, 3.2.1). Finally, we use the resulting slot labels in the corpus to train a neural slot tagger (Section 3.3).

1. *Selecting domain-relevant slots from candidates provided by weak supervision from domain-generic linguistic annotation tools.* We use FrameNet-style (Fillmore, 1976) semantic frames as our main source of weak supervision.[1] We also explore named entity recognition (NER).

2. *Training a standalone slot tagger for the selected slots.* Based on the discovered slots, we train a slot tagger to annotate in-domain utterances. After it is trained, the slot tagger can be used as a standalone component – it does not need the original annotation tools for prediction, and is able to improve on their results.

3. *Evaluation on multiple domains.* We show that our approach is domain-independent. We achieve state-of-the-art results for slot tagging without manual supervision in four different domains, with a 6-16% absolute F1 score increase over the previous benchmark.

4. *Downstream task application.* We evaluate our approach in a full dialogue response generation task. Our slots can be directly used to perform dialogue state tracking by merging annotations from consecutive turns. We train an end-to-end neural dialogue system using our automatically discovered slots in the restaurant domain and demonstrate that our approach improves performance over an unsupervised model, finding the correct venue in 5% more cases (35% more when no restaurant ontology is provided).

Our experimental code is available on GitHub.[2]

[1] See http://framenet.icsi.berkeley.edu/

[2] https://github.com/vojtsek/joint-induction

2 Related Work

The idea of using weak supervision to perform fine-grained language understanding based on domain-relevant (slot-like) attributes was proposed by Heck and Hakkani-Tür (2012), who construct a triple-based database of entity relations based on web search. They exploit the structure of in-domain web pages to obtain semantic annotations. There are also similar works on relation detection (Hakkani-Tür et al., 2013) or entity extraction (Wang et al., 2014). This approach is, however, limited by requiring structured web pages as underlying data.

Chen et al. (2014) combine semantic frame parsing with word embeddings for weakly supervised semantic slot induction. Chen et al. (2015) also use semantic frames, construct lexical knowledge graphs and perform a random walk to get slot candidates. However, both approaches only output a ranking of potential slot candidates based on frames. Since frame annotation is very fine-grained, this produces a huge number of candidates, requiring their manual merging into slots for any practical use. In contrast, we determine domain-relevant slots automatically. Coope et al. (2020) focus on a few-shot setting and perform span extraction of slot values using pretrained models. Their approach, however, still requires some expert annotation. Another direction of research focuses on zero-shot slot filling. Bapna et al. (2017)'s recurrent-neural-network-based slot tagger is pretrained on multiple domains and takes a textual description of the target slot on the input in addition to the user utterance. This way, adapting to a new domain only involves providing new slot descriptions. Further works extend this idea with more complex architectures (Shah et al., 2019; Liu et al., 2020).

Unsupervised and semi-supervised methods were also investigated for predicting intents (user

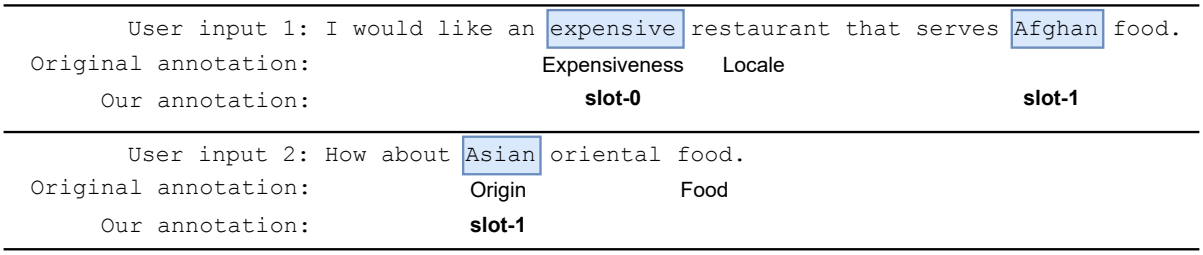

Figure 3: A sample of a dialogue from CamRest676 data, with labels from a frame-semantic parser (middle) and our slot tagger (bottom). Although "Afghan" food is not in the frame parser output, our tagger was able to recognize it. The change in value for slot-1 (corresponding to food type) is successfully captured in the second utterance. This shows that our model can categorize entities (both "Afghan" and "Asian" relate to the same slot).

input sentence types). Yang et al. (2014) use semi-supervised intent clustering, with manual annotation to seed and interpret the clusters. Chen et al. (2016) introduced a model for zero-shot intent embedding prediction based on similarity to known intents. Shi et al. (2018) proposed a fully unsupervised intent detection model with the use of sentence clustering based on sentence-level features.

Most applications of unsupervised or semi-supervised methods to end-to-end dialogue response generation avoid explicit dialogue state modeling (e.g., Serban et al., 2016; Li et al., 2016; Gao et al., 2019). They aim at a non-task-oriented setting, where state interpretability or response controllability are less of a concern. Other works in task-oriented dialogues use transfer learning for adapting to low-resourced target domains (Zhao and Eskenazi, 2018; Shalyminov et al., 2019), but also keep the dialogue state representation latent.

In contrast, Jin et al. (2018) propose to model the dialogue state explicitly, in a semi-supervised way. They extend the end-to-end encoder-decoder Sequicity model of Lei et al. (2018, cf. Section 4) by introducing an additional decoder that has access to posterior information about the system response. This allows them to train a state representation with a reconstruction loss on unsupervised examples, using the state as a limited memory for essential concepts (roughly corresponding to slots). Their method can be applied in fully unsupervised way, but it still requires some amount of in-domain annotations to achieve good performance. Our work aims at explicit dialogue state modeling without the need for any in-domain supervision.

3 Method

Our slot discovery method has three main stages: (1) We obtain weak supervision labels from automatic domain-generic annotation. (2) We identify domain-relevant slots based on the annotation labels by iteratively (a) merging and (b) ranking and selecting most viable candidates (Section 3.2). (3) we use the discovered slots to train an independent slot tagger (Section 3.3).

3.1 Acquiring labels

Figure 2 shows the overall data flow of our slot annotation pipeline. The data are first labeled with domain-generic linguistic annotation models, which we consider weak supervision. For our experiments, we use a frame semantic parser and NER, but other models, such as semantic role labeling (SRL; e.g., Palmer et al., 2010) or keyword extraction (e.g., Hulth, 2003) can be used in general. We use a simple union of labels provided by all annotation models.[3]

3.2 Discovering Slots: Merging and Ranking

Subsequent steps identify domain-relevant slots based on candidates provided by the automatic annotation. The slot discovery process is iterative – in each iteration, it: (1) merges similar candidates, (2) ranks candidates' relevance and eliminates irrelevant ones. Once no more frames are eliminated, the process stops and we obtain slot labels, which are used to train a slot tagger (see Section 3.3).

We refer to the automatically tagged tokens as *(slot) fillers*, and the tags are considered slot candidates. We use generic precomputed word embeddings as word representation in both steps. We further compute *slot embeddings* $e(s_k)$ for each distinct slot s_k as word embedding averages over

[3] If the same token is labeled multiple times by different annotation sources, both labels are considered candidates and are very likely to be merged. If multiple labels remain after the merging and ranking process, only the first label is kept, the rest are discarded.

all respective slot fillers, weighted proportionally by filler frequency. The slot embeddings need to be re-computed after each iteration due to the merging step. We will now describe the individual steps.

3.2.1 Candidate Merging

Since automatic annotation may have a very fine granularity,[4] entities/objects of the same type are often captured by multiple slot candidates. With a frame parser, for instance, the frames *Direction* and *Location* both relate to the concept of *area*. We thus need to merge similar $s_1 \dots s_n$ under a single candidate. We measure similarity of slots s_1, s_2 as:

$$\text{sim}(s_1, s_2) = \text{sim}_e(e(s_1), e(s_2)) + \text{sim}_{\text{ctx}}(s_1, s_2)$$

where sim_e is a cosine similarity and $\text{sim}_{\text{ctx}}(s_1, s_2)$ is a normalized number of occurrences of s_1 and s_2 with the same dependency relation. If the similarity exceeds a pre-set threshold T_{sim}, the candidates are merged into one.

3.2.2 Candidate Ranking and Selection

The main goal of this step is to remove irrelevant slot candidates and select the viable ones only. We hypothesize that different slots are likely to occur in different contexts (e.g., addresses are requested more often than stated by the user). To preserve relevant slots that only occur in rarer contexts, we cluster the data according to verb-slot pairs. We then rank candidates within each cluster (see details below). We consider candidates with a score higher than α-fraction of a given cluster mean to be relevant and select them for the next rounds. If a slot candidate is selected in at least one of the clusters, it is considered viable overall.

Clustering the data We process the data with a generic SRL tagger. Each occurrence of a filler is thus associated with a *head verb* whose semantic argument the corresponding word is, if such exists. We then compute embeddings of the formed *verb-filler* pairs as average of the respective token embeddings. The pairs are then clustered using agglomerative (bottom-up) hierarchical clustering with average linkage according to cosine distance of their embeddings.[5] The process stops when a predetermined number of clusters is reached.

Candidate Ranking criteria We use the following metrics to compute the ranking score:[6]

- **Frequency** $\text{frq}(s)$ is used since candidates that occur frequently in the data are likely important.
- **Coherence** $\text{coh}(s)$ is the average pairwise similarity of all fillers' embeddings:

$$\text{coh}(s) = \frac{\sum_{(v,w) \in C_s^2} d_{\cos}(e(v), e(w))}{|C_s^2|} \quad (1)$$

  where C_s^2 is a set of all pairs of fillers for the slot candidate s. We follow Chen et al. (2014)'s assumption that fillers with high coherence, i.e., focused on one topic, are good slot candidates.
- **TextRank** (Mihalcea and Tarau, 2004) is a keyword extraction algorithm. It constructs a graph where nodes represent words and edges represent their co-occurrence. The dominant eigenvector of the adjacency matrix of this graph then gives the individual words' scores. We replace fillers with candidate labels when computing the score, so we obtain results related to slots rather than to particular values.

The final score is a simple sum of rankings with respect to all three scores.

3.3 Slot Tagger Model Training

Our method described in Section 3.2 can give us a good set of dialogue slots. However, using the merged and filtered slots directly may result in low recall since the original annotation models used as weak supervision are not adapted to our specific domain. Therefore, we use the obtained labels to train a new, domain-specific slot tagger to improve performance. The tagger has no access to better labels than those derived by our method; however, it has a simpler task, as the set of target labels is now much smaller and the domain is much narrower.

We model the slot tagging task as sequence tagging, using a convolutional neural network that takes word- and character-based embeddings of the tokens as the input and produces a sequence of respective tags (Lample et al., 2016).[7] The output layer of the tagger network gives softmax probability distributions over possible tags. To further increase recall, we add an inference-time rule – if

[4] This is indeed the case for frame-semantic annotation, which we mostly use in our experiments in Section 5. Annotation types that have fewer label types could be further distinguished by e.g. adding the head verb from syntactic parsing, or using word classes/word clustering over the fillers.

[5] Note that fillers for the same slot candidate may end up in multiple clusters. This does not mean that the respective slot candidate is split – it is just ranked for relevance multiple times (with respect to multiple contexts).

[6] Usefulness of the individual metrics is confirmed in an ablation study in Section 6.

[7] https://github.com/deepmipt/ner

the most probable predicted tag is 'O' (i.e., no slot) and the second most probable tag has a probability higher than a preset threshold T_{tag}, the second tag is chosen as a prediction instead. As we discuss in Section 6, this threshold is crucial for achieving substantial recall improvement.

To improve the robustness of our model, we only use 10% of the original in-domain training set (with labels from Section 3.1) to train the slot tagger model. The rest of the training set is used for a grid search to determine model hyperparameters (hidden layer size, dropout rate and T_{tag} threshold). We choose the parameters that yield the best F1 score when compared against the automatic slot discovery results (i.e., no manual annotation is needed here, the aim is at good generalization).

4 Application in Dialogue Response Generation

To verify the usefulness of the labels discovered by our method, we use them to train and evaluate an end-to-end task-oriented dialogue system. We choose Sequicity (Lei et al., 2018) for our experiments, an LSTM-based encoder-decoder model that uses a system of copy nets and two-stage decoding. First, it decodes the dialogue state, so the database can be queried externally. In the subsequent step, Sequicity generates the system response conditioned on the belief state and database results. This architecture works with a flat representation of the dialogue state, i.e. the state is represented as a sequence of tokens – slot values.

The default Sequicity model uses gold-standard dialogue state annotation. However, a compatible state representation is directly obtainable from our labels, simply by concatenating the labels aggregated in each turn from user utterances. Whenever a new value for a slot is found in user input by our tagger, it is either appended to the state representation, or it replaces a previous value of the same slot. This artificial supervision thus allows us to provide a learning signal to the Sequicity model even without manually labeled examples.

5 Experiments

We evaluate our approach to slot discovery by comparing the resulting slot labels to gold-standard supervised slot annotation. Additionally, we evaluate the structure of clusters created during the selection process (Section 3.2.2) by comparing it to gold-standard user intents. We also test the usefulness of our labels in a full dialogue response generation setup (Section 4), where we compare to gold-standard dialogue tracking labels.

5.1 Datasets and Experimental Setup

We use the following datasets for our experiments:

- **CamRest676** (**CR**) (Wen et al., 2017) has 676 dialogues, 2,744 user utterances, 4 tracked slots and 2 intents in the restaurant domain.
- **MultiWOZ** (Budzianowski et al., 2018; Eric et al., 2020) is a multi-domain corpus; we picked two domains – hotel reservation and attraction recommendation – to form **WOZ-hotel** (**WH**) with 14,435 utterances, 9 slots, 3 intents and **WOZ-attr** (**WA**) with 7524 utterances, 8 slots and 3 intents respectively.[8]
- **Cambridge SLU** (Henderson et al., 2012) (**CS**) contains 10,569 utterances and tracks 5 slots with 5 intents in the restaurant domain.
- **ATIS** (**AT**) (Hemphill et al., 1990) contains 4,978 utterances with 79 slots and 17 intents in the flights domain.[9]

As sources of weak supervision providing slot candidates, we mainly use the frame semantic parsers *SEMAFOR* (Das et al., 2010) and *open-sesame* (Swayamdipta et al., 2017) – a union of labels provided by both parsers is used in all our setups. In addition, to explore combined sources on the named-entity-heavy ATIS dataset, we include a generic convolutional NER model provided by SpaCy.[10] To provide features for slot candidate merging and selection, we use AllenNLP (Gardner et al., 2017) for SRL and FastText (Bojanowski et al., 2017) as pretrained word embeddings.

Slot merging and selection parameters were set heuristically in an initial trial run on the CamRest676 data and proved stable across domains. Slot tagger hyperparameters are chosen according to grid search on a portion of the training data, as described in Section 3.3.[11]

5.2 System Variants and Baselines

We test multiple ablation variants of our method:

- *Ours-full* is the full version of our method (full annotation setup and trained slot tagger).

[8] MultiWOZ contains more domains such as *restaurant, train search, bus search*. However, we decided to not include these as they are nearly identical to the other domains we use.

[9] We used the ATIS data version from https://www.kaggle.com/siddhadev/atis-dataset-from-ms-cntk.

[10] https://spacy.io

[11] Training details are included in Appendix C.

method ↓ / dataset→	**CR**	**CS**	**WH**	**WA**	**AT**	
Tag-supervised*	**0.778** ± .004	0.724 ± .003	**0.742** ± .008	**0.731** ± .002	**0.848** ± .003	
Dict-supervised*	0.705 ± .005	**0.753** ± .005	**0.750** ± .018	0.665 ± .003	0.678 ± .002	
weak supervision →	frames	frames	frames	frames	frames	frames & NER
Chen et al.	0.535 ± .002	0.590 ± .001	0.382 ± .001	0.375 ± .001	0.616 ± .001	–
Ours-nocl	0.311 ± .006	0.393 ± .011	0.122 ± .001	0.266 ± .008	0.631 ± .002	0.677 ± .002
Ours-notag	0.552 ± .008	0.664 ± .007	0.388 ± .002	0.383 ± .002	0.627 ± .002	0.648 ± .003
Ours-nothr	0.586 ± .024	0.569 ± .031	0.485 ± .032	0.435 ± .002	0.671 ± .005	0.698 ± .004
Ours-full	**0.665** ± .012	**0.692** ± .008	**0.548** ± .004	**0.439** ± .001	0.678 ± .002	**0.710** ± .002

Table 1: F1 score values with 95% confidence intervals for slot tagging performance comparison among different methods (see Section 5.2). The respective precision and recall values are presented in the Appendix (Table 7). The measures are evaluated using a manual slot mapping to the datasets' annotation, which is not needed for the methods themselves (see Section 5.3). *Note that supervised setups are not directly comparable to our approach.

- *Ours-nothr* does not use the recall-increasing second-candidate rule in the slot tagger (cf. Section 3.3).
- *Ours-notag* excludes the slot tagger, directly using the output of our merging and selection step.
- *Ours-nocl* further excludes the clustering step; slot candidate ranking and selection is performed over all candidates together (cf. Section 3.2.2).

We also compare to previous work of Chen et al. (2014),[12] which is similar to *Ours-nocl*, but it does not merge similar frames and uses different ranking criteria. To put our results into perspective, we also include two supervised models for comparison: *Tag-supervised* is the same model that we use as our slot tagger (see Section 3.3), but it is trained on supervised data. *Dict-supervised* uses a simple dictionary of labels obtained from the training data.

As an intrinsic evaluation of the verb-slot pair clusters formed for slot ranking in Section 3.2.2, we compare to gold-standard intent annotation with respect to the following baselines: (1) a majority baseline (assigning the most frequent intent class to all instances), and (2) a simple method that represents the utterances as averages of respective word embeddings and performs sentence-level intent clustering. All the slots in a given utterance are then assumed to have the same intent.

The dialogue generation task is evaluated by comparing to Jin et al. (2018)'s approach introduced in Section 2. We run their model in a fully unsupervised way, i.e. we provide no labeled examples during the training phase, to give a fair comparison against our model. To provide more perspective, we also show a supervised variant of Jin et al. (2018)'s model, where gold-standard slot labels are provided.

[12]We use our own reimplementation of their approach.

5.3 Evaluation Metrics

For evaluation, we construct a handcrafted *reference mapping* between our discovered slots and the respective ground-truth slots and intents. The mapping is domain-specific, but it is very easy to construct even for an untrained person – the process takes less than 10 minutes for each of our domains. It amounts to matching slots from the domain ontology against slots output by our approach, which are represented by FrameNet labels. Most importantly, the mapping is ***only needed for evaluation***, not by our method itself. We provide an example mapping in Appendix B.

We use the following evaluation metrics:

- **Slot F1 score**: To reflect slot tagging performance, we measure precision, recall, and F1 for every slot individually. An average is then computed from slot-level scores, weighted by the number of slot occurrences in the data. We measure slot F1 both on standalone user utterances (slot tagging) and in the context of a dialogue system (dialogue tracking).
- **Slot-level Average Precision (AP)**. The slot candidates picking task is a ranking problem and we use the *average precision* metric following Chen et al. (2014). Considering a ranked list of discovered slots $l = s_1, \ldots, s_k, \ldots, s_n$ we compute AP:

$$AP(l) = \frac{\sum_{k=1}^{n} P@k(l)\mathbb{1}_k}{\#\,\text{mapped slots}} \qquad (2)$$

  where $\mathbb{1}_k$ is an indicator function that equals one if slot k has a reference mapping defined and $P@k(l)$ is precision at k of the ranked list l.
- **Slot Rand Index (RI)** is a clustering metric, used to evaluate slot candidate merging. RI is the proportion of pairs of slot candidates that are correctly assigned into the same or into different

method	CR	CS	WH	WA	AT
Chen et al.	0.315 ±.002	0.272 ±.001	0.269 ±.001	0.393 ±.002	**0.267** ±.003
Ours-nocl	**0.519** ±.003	0.376 ±.003	0.069 ±.074	0.176 ±.016	0.069 ±.008
Ours-full	**0.520** ±.004	**0.400** ±.003	**0.317** ±.008	**0.403** ±.006	0.208 ±.018

Table 2: Slot candidate ranking average precision for all datasets (see Sections 5.2 and 5.3 for details).

	method	CR	CS	WH	WA	AT
RI	Rnd	0.466	0.268	0.155	0.153	0.178
	Ours	0.587	0.319	0.168	0.188	0.171
NMI	Rnd	0.212	0.137	0.061	0.128	0.171
	Ours	0.359	0.207	0.101	0.117	0.194

Table 3: Slot merging evaluation using RI and NMI (cf. Section 5.3) on selected datasets, comparing our approach (*Ours*) with a random baseline (*Rnd*).

slots (following the reference mapping).[13]

- **Normalized Mutual Information (NMI)** is the mutual information between two clusterings normalized into the (0, 1) interval. Thanks to the normalization, it is suitable for comparing two clusterings with different numbers of clusters.
- **Intent Accuracy** is the percentage of slot occurrences assigned into the correct intent cluster under the reference mapping (see Section 5.2).
- **Dialogue Joint Goal Accuracy** calculates the proportion of dialogue turns where all user constraints (i.e., dialogue state summarizing slot values) are captured correctly (Mrkšić et al., 2017).
- **Dialogue Entity Match Rate** calculates the last turn's entity in each dialogue. It verifies if a correct entity would be retrieved from the database using the final constraints (Wen et al., 2017).

For slot tagging and ranking evaluation, we sampled a random data order 50 times and performed 5-fold cross-validation for each permutation. For the dialogue generation evaluation, we trained the models 100 times and used averaged results. All results are given with 95% confidence intervals.

6 Results and Discussion

We first evaluate the main task of slot tagging and include a manual error analysis, then present detailed results for subtasks (slot candidate ranking and merging) and additional tasks (intent clustering and full response generation).

Slot tagging is evaluated in Table 1. *Ours-full* (slot selection + trained tagger) outperforms all other approaches by a large margin, especially in terms of recall. The performance cannot match the supervised models, but it is not far off in some domains.[14] Chen et al. (2014)'s method has a slightly higher precision, but our recall is much higher than theirs (see Appendix A.1). Note that Chen et al. (2014) do not reduce the set of candidates, they only rank them so that a manual cut-off can be made. In contrast, our method reduces the set of candidates significantly. A comparison between *Ours-notag* and *Ours-full* shows that applying the slot tagger improves both precision and recall. Tagger without the threshold decision rule (*Ours-nothr*) mostly performs better than the parser; however, using the threshold is essential to improve recall. Experiments on ATIS with NER as an additional source of annotation proved that our method can benefit from it. As discussed above, the use of the trained tagging model is crucial to improve the recall of our method. In Figure 4, we compare the results with and without the tagger. We change the value of prediction threshold and measure the number of cases in which the tagging model encounters more true positives, false positives or false negatives, respectively. As the results show, lowering the threshold increases the number of cases in which the tagger finds more correct slot values (and therefore improves recall), while it does not affect the number of false positives much (and therefore retains precision).

Error analysis: We conducted a manual error analysis of slot tagging to gain more insight about the output quality and sources of errors. In general, we found that the tagger can generalize and capture unseen values (cf. Figure 3).

One source of errors is the relatively low recall of the frame-semantic parsers used. We successfully address this issue by introducing the slot tagger, however, many slot values remain untagged. This is expected as our method's performance is inherently limited by the input linguistic annotation quality. Another type of errors is caused by the can-

[13] We compute RI on a union of labels that have a ground-truth slot mapping and all labels selected by our method. Labels without ground-truth mapping are assumed to form single-item "pseudo-slots".

[14] Note that our measurements of slot F1 only consider the 'O' tag as negative (the average is computed over slots only). This results in lower numbers than those reported in literature (cf. e.g. Goo et al., 2018), but we believe that this reflects the actual performance more accurately.

[15] We present results taken in unsupervised setting, i.e. when no ontology is available. However, since Jin et al. (2018) consider only slot values that are known from the ontology by default, we provide the extended results in Appendix A.2.

method	Slot F1	Joint Goal Accuracy	Entity Match Rate
Jin et al. supervised	0.967 ± .001	0.897 ± .002	0.869 ± .004
Jin et al. unsupervised	0.719 ± .002	0.385 ± .003	0.019 ± .002
Jin et al. weak-labels	0.709 ± .011	0.335 ± .008	0.269 ± .012
Ours-full (unsupervised)	**0.756** ± .004	**0.465** ± .007	**0.368** ± .008

Table 4: Evaluation on the downstream task of dialogue generation on CamRest676 data. We evaluate with respect to three state tracking metrics (see Section 5.3). The best results in an unsupervised setting are presented in bold.[15]

method	CR	CS	WH	WA	AT
Majority	0.592	0.530	0.883	0.612	**0.727**
Embedding	0.535	0.551	0.873	0.595	0.705
Ours	**0.705**	**0.613**	**0.882**	**0.699**	0.677

Table 5: Cluster assignment accuracy of our methods if we interpret the clustering as user intent detection. *Majority* is a majority baseline and *Embedding* refers to an average sentence embedding clustering approach.

configuration	F1 score
Ours-full	**0.663** ± 0.012
Ours -frq	0.600 ± 0.008
Ours -coh	0.582 ± 0.012
Ours -TextRank	0.514 ± 0.006

Table 6: Ablation study of slot ranking features on CamRest676. The full model is compared to variants leaving out of the scores described in Section 3.2.2.

didate merging procedure (see also below). Due to frequent co-occurrence, it might happen that two semantically unrelated candidates are merged and therefore some tokens are wrongly included as respective slot fillers. Nevertheless, the merging step is required in order to obtain a reasonable number of slots for a dialogue domain.

Our approach does leave some room for improvements, especially regarding the consistency of results across different slots, which can be imbalanced. For instance, on the WOZ-hotel data, we observe a difference of up to 0.5 F1 score among individual slots (see Appendix A.2).

Slot candidate ranking results are given in Table 2. Our pipeline significantly outperforms Chen et al. (2014)'s approach on 4 out of 5 datasets. We can also see that the slot-verb pairs clustering step is important – in the ablation experiment where we do not perform clustering (*Ours-nocl*), performance falls dramatically on the WOZ-hotel, WOZ-attr and ATIS data. This is because without the clustering step, a large number of context-irrelevant slot candidates is considered, hurting performance.

In addition, we include a detailed evaluation of the contribution of the individual slot candidate ranking scores described in Section 3.2.2. Results in Table 6 suggest that all of our proposed scores improve the performance.

Slot merging evaluation is shown in Table 3. Although candidates in the CamRest676 data are merged into slots reasonably well, other datasets show a relatively low performance. The low RI scores are a result of errors in candidate ranking, which wrongly assigned high ranks to some rare, irrelevant candidates. These candidates do not appear in the reference mapping and are assumed to form singular "pseudo-slots". However, they are typically joined with similar candidates in the merging process. This leads to many pairs of candidates that are merged into one slot by our approach but appear separately in the reference mapping. Nevertheless, this behavior barely influences slot tagging performance as the candidates are rare.

Clustering evaluation: Table 5 suggests that our clustering performs better than simple baselines and can potentially yield useful results if used for intent detection. Nevertheless, intent detection is more complex and presumably requires more features and information about the dialogue context, which we reserve for future work. The complexity is also suggested by the fact that the naive embedding clustering performs worse than the majority baseline in 4 out of 5 cases.

Dialogue response generation: We explore the influence that our labels have on sequence-to-sequence dialogue response generation in an experiment on the CamRest676 data (see Table 4). We can see that our method provides helpful slot labels that improve dialogue state tracking performance. Compared to Jin et al. (2018)'s system used in a fully unsupervised setting, our approach shows significant improvements in all metrics. We achieve better results than Jin et al. (2018)'s system especially with respect to entity match rate, suggesting that our model can provide consistent labels throughout the whole dialogue. To make a fair comparison, we further evaluate Jin et al. (2018)'s system in a setting in which it can learn

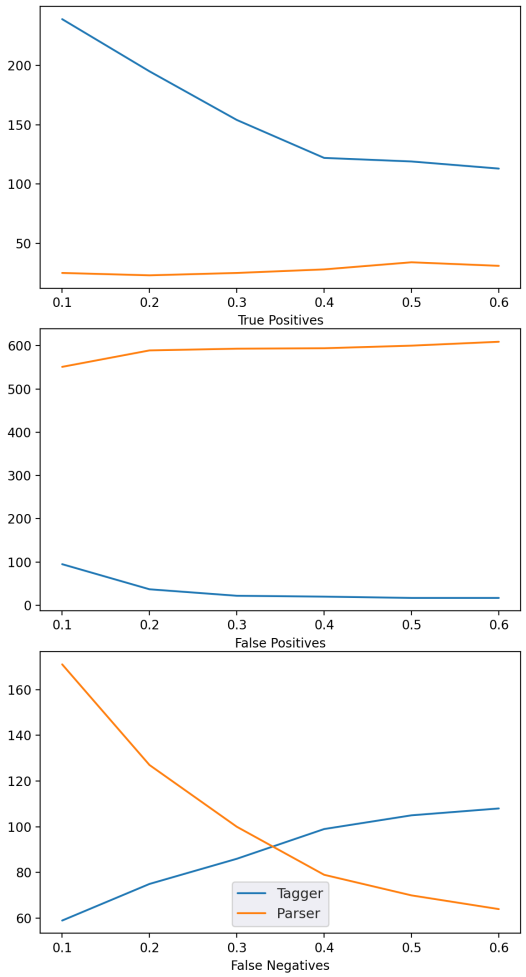

Figure 4: The comparison of outputs of our tagger and the parser. The plots show a number of cases in which the respective approach encounters more TPs, FPs or FNs than the other.

from the labels provided directly by weak supervision (i.e., the frame-semantic parser, not filtered by our pipeline). We observe an improvement in terms of entity match rate, but it does not match the improvement achieved with our filtered labels. Surprisingly, slot F1 and joint goal accuracy even decrease slightly, which suggests that label quality is important and the noisy labels obtained directly from weak supervision are not useful enough.

7 Conclusion

We present a novel approach for weakly supervised natural language understanding in dialogue systems that discovers domain-relevant slots and tags them in a standalone fashion. Our method removes the need for annotated training data by using off-the-shelf linguistic annotation models. Experiments on five datasets in four domains mark a significant improvement in intrinsic NLU performance over previous weakly supervised approaches; in particular, we vastly improve the slot recall. The usefulness of slots discovered by our method is further confirmed in a full dialogue response generation application. Code used for our experiments is available on GitHub.[16]

A drawback of our approach is the reliance on existing linguistic annotation models. We show that the method is able to combine multiple annotation sources and create a tagger that functions as a standalone component, generalizing better than the original annotation and thus lowering this dependency. Nevertheless, the results are still somewhat limited by the input annotation structure and quality. In future, we plan to further improve the model by unsupervised selection of slot candidates via keyword extraction and clustering, as well as by taking context information from preceding dialogue turns into account. We also want to focus more on the intent detection aspect of our work.

Acknowledgements

This work was supported by Charles University grants PRIMUS 19/SCI/10, GAUK 302120, and SVV 260 575. We also want to thank Jindřich Libovický and David Mareček for helpful comments on the draft, and the anonymous reviewers for their remarks that helped us further improve the paper.

References

Ankur Bapna, Gokhan Tür, Dilek Hakkani-Tür, and Larry Heck. 2017. Towards Zero-Shot Frame Semantic Parsing for Domain Scaling. In *Proceedings of Interspeech 2017*, pages 2476–2480.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the ACL*, 5:135–146.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Inigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. MultiWOZ – a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling. In *Proceedings of EMNLP*.

Yun-Nung Chen, Dilek Hakkani-Tür, and Xiaodong He. 2016. Zero-shot learning of intent embeddings for expansion by convolutional deep structured semantic models. In *Proceedings of IEEE ICASSP*, pages 6045–6049.

[16] https://github.com/vojtsek/joint-induction

Yun-Nung Chen, William Yang Wang, and Alexander Rudnicky. 2015. Jointly modeling inter-slot relations by random walk on knowledge graphs for unsupervised spoken language understanding. In *Proceedings of NAACL*, pages 619–629.

Yun-Nung Chen, William Yang Wang, and Alexander I Rudnicky. 2014. Leveraging frame semantics and distributional semantics for unsupervised semantic slot induction in spoken dialogue systems. In *Proceedings of IEEE SLT*, pages 584–589.

Samuel Coope, Tyler Farghly, Daniela Gerz, Ivan Vulić, and Matthew Henderson. 2020. Span-ConveRT: Few-shot Span Extraction for Dialog with Pretrained Conversational Representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 107–121, Online.

Dipanjan Das, Nathan Schneider, Desai Chen, and Noah A. Smith. 2010. Probabilistic frame-semantic parsing. In *Proceedings of NAACL-HLT*, pages 948–956, Los Angeles, California.

Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyang Gao, Adarsh Kumar, Anuj Goyal, Peter Ku, and Dilek Hakkani-Tur. 2020. MultiWOZ 2.1: A Consolidated Multi-Domain Dialogue Dataset with State Corrections and State Tracking Baselines. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 422–428, Marseille, France.

Charles J Fillmore. 1976. Frame semantics and the nature of language. *Annals of the New York Academy of Sciences*, 280(1):20–32.

Xiang Gao, Sungjin Lee, Yizhe Zhang, Chris Brockett, Michel Galley, Jianfeng Gao, and Bill Dolan. 2019. Jointly Optimizing Diversity and Relevance in Neural Response Generation. In *Proceedings of NAACL*, Minneapolis, MN, USA.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of ACL Workshop for NLP Open Source Software (NLP-OSS)*.

Chih-Wen Goo, Guang Gao, Yun-Kai Hsu, Chih-Li Huo, Tsung-Chieh Chen, Keng-Wei Hsu, and Yun-Nung Chen. 2018. Slot-Gated Modeling for Joint Slot Filling and Intent Prediction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 753–757, New Orleans, Louisiana.

Dilek Hakkani-Tür, Larry Heck, and Gokhan Tur. 2013. Using a knowledge graph and query click logs for unsupervised learning of relation detection. In *Proceedings of IEEE ICASSP*, pages 8327–8331.

Larry Heck and Dilek Hakkani-Tür. 2012. Exploiting the semantic web for unsupervised spoken language understanding. In *Proceedings of IEEE SLT*, pages 228–233.

Charles T. Hemphill, John J. Godfrey, and George R. Doddington. 1990. The ATIS spoken language systems pilot corpus. In *Proceedings of the workshop on Speech and Natural Language - HLT '90*, pages 96–101, Hidden Valley, Pennsylvania.

Matthew Henderson, Milica Gašić, Blaise Thomson, Pirros Tsiakoulis, Kai Yu, and Steve Young. 2012. Discriminative spoken language understanding using word confusion networks. In *Proceedings of IEEE SLT*, pages 176–181.

Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of EMNLP*, pages 216–223.

Xisen Jin, Wenqiang Lei, Zhaochun Ren, Hongshen Chen, Shangsong Liang, Yihong Zhao, and Dawei Yin. 2018. Explicit state tracking with semi-supervision for neural dialogue generation. In *Proceedings of ACM CIKM*, pages 1403–1412.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL*, pages 260–270.

Wenqiang Lei, Xisen Jin, Min-Yen Kan, Zhaochun Ren, Xiangnan He, and Dawei Yin. 2018. Sequicity: Simplifying task-oriented dialogue systems with single sequence-to-sequence architectures. In *Proceedings of ACL*, pages 1437–1447.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A Diversity-Promoting Objective Function for Neural Conversation Models. In *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, San Diego, CA, USA.

Zihan Liu, Genta Indra Winata, Peng Xu, and Pascale Fung. 2020. Coach: A Coarse-to-Fine Approach for Cross-domain Slot Filling. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 19–25, Online.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of EMNLP*, pages 404–411.

Nikola Mrkšić, Diarmuid O Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2017. Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of ACL*, pages 1777–1788.

Martha Palmer, Daniel Gildea, and Nianwen Xue. 2010. Semantic role labeling. *Synthesis Lectures on Human Language Technologies*, 3(1):1–103.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of AAAI*, pages 3776–3783.

Darsh Shah, Raghav Gupta, Amir Fayazi, and Dilek Hakkani-Tur. 2019. Robust Zero-Shot Cross-Domain Slot Filling with Example Values. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5484–5490, Florence, Italy.

Igor Shalyminov, Sungjin Lee, Arash Eshghi, and Oliver Lemon. 2019. Few-shot dialogue generation without annotated data: A transfer learning approach. In *Proceedings of SIGDIAL*, Stockholm, Sweden.

Chen Shi, Qi Chen, Lei Sha, Sujian Li, Xu Sun, Houfeng Wang, and Lintao Zhang. 2018. Auto-Dialabel: Labeling dialogue data with unsupervised learning. In *Proceedings of EMNLP*, pages 684–689.

Swabha Swayamdipta, Sam Thomson, Chris Dyer, and Noah A Smith. 2017. Frame-semantic parsing with softmax-margin segmental RNNs and a syntactic scaffold. *arXiv:1706.09528*.

Lu Wang, Larry Heck, and Dilek Hakkani-Tür. 2014. Leveraging semantic web search and browse sessions for multi-turn spoken dialog systems. In *Proceedings of IEEE ICASSP*, pages 4082–4086.

Tsung-Hsien Wen, David Vandyke, Nikola Mrksić, Milica Gašić, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of EACL*, pages 438–449.

Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2013. The dialog state tracking challenge. In *Proceedings of SIGDIAL*, pages 404–413.

Xiaohao Yang, Jia Liu, Zhenfeng Chen, and Weilan Wu. 2014. Semi-supervised learning of dialogue acts using sentence similarity based on word embeddings. In *Proceedings of ICALIP*, pages 882–886.

S. Young, M. Gasic, B. Thomson, and J.D. Williams. 2013. POMDP-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

Tiancheng Zhao and Maxine Eskenazi. 2018. Zero-shot dialog generation with cross-domain latent actions. In *Proceedings of SIGDIAL*, pages 1–10.

A Additional Results

A.1 Slot tagging precision and recall

method	CR		CS		WH		WA		AT		AT(+NER)	
	P	R	P	R	P	R	P	R	P	R	P	R
Tag-supervised*	0.794	0.814	0.823	0.696	0.880	0.683	0.802	0.715	0.772	0.913	–	–
Dict-supervised*	0.793	0.710	0.831	0.752	0.869	0.710	0.669	0.859	0.546	0.990	–	–
Chen et al.	**0.771**	0.486	**0.813**	0.529	0.384	0.579	0.362	0.462	0.701	0.583	–	–
Ours-nocl	0.537	0.347	0.616	0.371	0.101	0.218	0.244	0.340	0.634	0.595	0.662	0.704
Ours-notag	0.561	0.586	0.690	0.688	0.369	0.607	0.335	0.575	0.715	0.642	0.685	0.623
Ours-nothr	0.636	0.549	0.585	0.566	0.458	0.575	**0.394**	0.561	0.701	0.687	**0.710**	0.697
Ours-full	0.752	**0.643**	0.718	**0.703**	**0.494**	**0.750**	0.373	**0.606**	0.684	0.672	0.703	**0.725**

Table 7: Precision (P) and recall (R) values slot tagging performance comparison among different methods (see Section 5.2; frames are used as weak supervision in all setups, the rightmost column on ATIS additionally uses NER). We can see consistent recall improvement when using our slot tagger. The measures are evaluated using a manually designed slot mapping to the datasets' annotation, which is not needed for the methods themselves (see Section 5.3). *Note that supervised setups are not directly comparable to our approach.

A.2 Individual slot performance

dataset	price	area	request	type	food	day	people	stars	stay
CR	0.543	0.764	0.759	–	0.590	–	–	–	–
CS	0.629	0.835	0.480	0.813	0.642	–	–	–	–
WH	0.208	0.524	0.107	0.125	–	0.146	0.822	0.821	0.341

Table 8: Per-slot F1 scores of the *Ours-full* method evaluated on selected datasets with slot intersection. For some slots the performance varies a lot among datasets due to different ranges of values and contexts. The measures are evaluated using a manually designed slot mapping to the datasets' annotation, which is not needed for the methods themselves (see Section 5.3).

method	Slot F1		Joint Goal Accuracy		Entity Match Rate	
	onto	no-onto	onto	no-onto	onto	no-onto
Jin et al. supervised	0.969 ± .001	0.967 ± .001	0.911 ± .002	0.897 ± .002	0.892 ± .004	0.869 ± .004
Jin et al. unsupervised	**0.873** ± .003	0.719 ± .002	**0.632** ± .009	0.385 ± .003	0.398 ± .010	0.019 ± .002
Ours-full (unsupervised)	0.821 ± .004	**0.756** ± .004	0.533 ± .007	**0.465** ± .007	**0.445** ± .009	**0.368** ± .008

Table 9: Evaluation on the downstream task of dialogue generation on CamRest676 data. We evaluate with respect to three distinct metrics of state tracking performance. Two variations of metrics are included: *onto* takes only slot values present in ontology into account, *no-onto* does not require ontology information and thus fits the unsupervised setting better (cf. Section 5.3). In bold we present the best results in unsupervised setting.

B Reference mapping

***Ours-full* output**		**CambridgeSLU ontology**
Expensiveness	↦	Pricerange
Origin + People_by_origin	↦	Food
Direction + Part_orientational	↦	Area
Contacting + Artifact	↦	Phone
Locale_by_use	↦	Type

Table 10: An example of reference mapping between the output of *Ours-full* represented by FrameNet labels (left) and ground-truth CambridgeSLU ontology (right). Frames merged by our method are shown on a single line, separated by "+".

C Training details

Here we provide details about training process and model sizes:

- Since the models are rather small with regards to number of parameters, it is sufficient to use a regular desktop PC. In our experiments, we require about 4 GB of RAM, and we use Intel Xeon E5-2630 v4 CPUs.
- Our slot candidate selection step takes roughly 1 hour. The tagger model is lightweight, with only 150k parameters. Its training requires 10-30 minutes, depending on the exact configuration and data size.
- The evaluation scripts are attached and described in the README file.
- We conduct hyperparameter search using a basic grid search algorithm. We tested hidden size values $\in [50, 200]$, dropout $\in [0.5, 0.85]$ and the threshold $T_{\text{tag}} \in [0.05, 0.3]$. Therefore, we ran $4 \times 8 \times 6 = 192$ search trials.
- The best parameters were determined by tagger accuracy on the validation set: hidden_size = 250, dropout = 0.7, $T_{\text{tag}} = 0.3$, $T_{\text{sim}} = 0.9$.
- Links to the data are included in the README file, we use *train:validation:split* ratio equal to *8:1:1*.