spacebook-project.eu

Report on learning semantic analysis components

Andreas Vlachos, Stephen Clark

Distribution: Public

SpaceBook

Spatial & Personal Adaptive Communication Environment: Behaviors & Objects & Operations & Knowledge

270019 Deliverable 4.1.3

February 28, 2014

Project funded by the European Community
under the Seventh Framework Programme for
Research and Technological Development

The deliverable identification sheet is to be found on the reverse of this page.

Project ref. no.	270019
Project acronym	SpaceBook
Project full title	Spatial & Personal Adaptive Communication Environment: Behaviors & Objects & Operations & Knowledge
Instrument	STREP
Thematic Priority	Cognitive Systems, Interaction, and Robotics
Start date / duration	01 March 2011 / 36 Months

Security	Public
Contractual date of delivery	M35 = January 2014
Actual date of delivery	February 28, 2014
Deliverable number	4.1.3
Deliverable title	Report on learning semantic analysis components
Type	Report
Status & version	Final 1.0
Number of pages	15 (excluding front matter)
Contributing WP	4
WP/Task responsible	UCAM
Other contributors	
Author(s)	Andreas Vlachos, Stephen Clark
EC Project Officer	Franco Mastroddi
Keywords	Semantic Parsing, Structured Machine Learning

The partners in SpaceBook are:	**Umeå University**	UMU
	University of Edinburgh HCRC	UE
	Heriot-Watt University	HWU
	Kungliga Tekniska Högskola	KTH
	Liquid Media AB	LM
	University of Cambridge	UCAM
	Universitat Pompeu Fabra	UPF

For copies of reports, updates on project activities and other SPACEBOOK-related information, contact:

The SPACEBOOK Project Co-ordinator:
Dr. Michael Minock
Department of Computer Science
Umeå University
Sweden 90187
mjm@cs.umu.se
Phone +46 70 597 2585 - Fax +46 90 786 6126

Copies of reports and other material can also be accessed via the project's administration homepage, `http://www.spacebook-project.eu`

Contents

Executive Summary **1**

1 Introduction **2**

2 Meaning Representation Language **2**

2.1 Predicates 4

3 Semantic parsing for the new corpus **4**

4 Task decomposition **5**

5 Imitation learning for structured prediction **8**

5.1 DAgger 8

5.2 Handling missing labels 10

6 Experiments **11**

7 Discussion **12**

8 Conclusions, Difficulties and Future Work **13**

Executive summary

This document describes the development of the machine-learned semantic parser, using the manually annotated corpus described in Deliverable 4.1.2 as training data. Semantic parsing is the task of translating natural language (NL) utterances into a machine-interpretable meaning representation (MR). Most approaches to this task have been developed and evaluated on a small number of existing corpora. While these corpora have made progress in semantic parsing possible, most of them cover rather narrow domains and context is rarely considered. Hence the SPACEBOOK project has provided an excellent opportunity to extend existing work by considering the more challenging domain—namely tourism related activities in Edinburgh—that SPACEBOOK offers.

One of the difficulties in treating the creation of a semantic parser as a machine learning problem is that the task of predicting a meaning representation for an utterance—particularly when the MR is as complex as that used in SPACEBOOK—is a highly complex operation. Hence traditional machine learning approaches relying on $\arg\max$ operations, for example the commonly used structured perceptron, cannot be applied. Instead, we chose to adapt the imitation learning algorithm DAGGER to learn a joint semantic parsing model. Our results, which are based on an intrinsic evaluation using a held-out portion of the corpus as a test set, suggest that the task defined by the new SPACEBOOK corpus is feasible, albeit more challenging than the semantic parsing tasks defined by existing corpora. The results also demonstrate the potential of applying imitation learning to complex structured prediction tasks.

The report concludes with a description of the difficulties experienced in building the semantic parser in time for integration into the full SPACEBOOK system, and some suggestions for future work.

1 Introduction

This report follows on from Deliverable 4.1.1 (Initial Request Analysis Component) in which the semantic parsing problem was formulated and an initial rule-based semantic parser was described, and Deliverable 4.1.2 (Final Request Analysis Component), which described the creation of the semantic parsing corpus and an initial attempt at building a machine-learned dialog act tagger (the first component in the semantic parser). Some of the introductory material from the earlier deliverables is repeated here, to make this report more self-contained.

The goal of this part of the project was to automatically learn a semantic grammar and parser from training data, where the data consists of pairs of SPACEBOOK-type utterances and their corresponding logical forms. The SPACEBOOK-type utterances are transcriptions of real speech captured during Wizard-of-Oz experiments, with the intention that these will be close to the output of the speech recognizer used in the SPACEBOOK-system. The logical forms are represented using the Meaning Representation Language (MRL) developed for SPACEBOOK, which was described in detail in Deliverable 4.1.1.

The semantic parsing corpus is a substantial contribution in its own right, based on two dialog scenarios and consisting of 17 dialogs and 2,374 user utterances, each utterance manually annotated with its semantic representation using the SPACEBOOK MRL. Compared to supervised syntactic parsing problems, e.g. the Penn Treebank parsing task for which there is roughly 40,000 annotated newspaper sentences [1], 2,374 utterances is relatively small. But compared to existing semantic parsing corpora, e.g. the ATIS [2] and GeoQuery [3] corpora, our new corpus is at least as large. However, given the increased complexity of the SPACEBOOK semantic parsing problem compared to earlier semantic parsing tasks, we still expected the machine learning task to be a challenging one. The earlier results on the dialog act tagging task, reported in Deliverable 4.1.2, suggested that the task is indeed difficult but not infeasible. Our new results reported here confirm this initial judgement.

In the remainder of this report, Section 2 repeats some of the description of the MRL from an earlier deliverable. Section 3 describes how semantic parsing can be applied to the new corpus, and Section 4 shows how the prediction of the MRL for an utterance can be broken down into a number of individual decisions. Section 5 describes imitation learning – a type of machine learning which we argue is particularly well-suited to the prediction of complex MRLs. Section 6 presents some experimental results based on an intrinsic evaluation using held-out sections of the corpus as test data. Finally, the concluding sections have some general discussion about the task and the difficulties experienced in the context of the SPACEBOOK project.

2 Meaning Representation Language

The MRL uses a flat syntax composed of elementary predications, based loosely on minimal recursion semantics [4], but without an explicit treatment of scope. Each meaning representation (MR) consists of a dialog act representing the overall function of the utterance, followed for some dialog acts by an unordered set of predicates. All predicates are implicitly conjoined and the names of their arguments specified to improve readability and to allow for some of the arguments to be optional. The argument values can be either constants from the controlled vocabulary, verbatim string extracts from the utterance (enclosed in quotes) or variables (`Xno`). Negation is denoted by a tilde ($\sim$) in front of predicates. The variables are used to bind together the arguments of different predicates within an utterance, as well as to denote coreference across utterances. Figure 1 contains part of an example dialog annotated with the MRL.

USER what's the nearest italian, em, for a meal?

```
dialogAct(set_question)
*isA(id:X1, type:restaurant)
def(id:X1)
hasProperty(id:X1, property:cuisine,
            value:"italian")
distance(location:@USER,
         location:X1, value:X2)
argmin(argument:X1, value:X2)
```

WIZARD vapiano's.

```
dialogAct(inform)
isA(id:X4, type:restaurant)
*isNamed(id:X4, name:"vapiano's")
equivalent(id:X1, id:X4)
```

USER take me to vapiano!

```
dialogAct(set_question)
*route(from_location:@USER,
       to_location:X4)
isA(id:X4, type:restaurant)
isNamed(id:X4, name:"vapiano")
```

WIZARD keep walking straight down clerk street.

```
dialogAct(instruct)
*walk(agent:@USER, along_location:X1,
      direction:forward)
isA(id:X1, type:street)
isNamed(id:X1, name:"clerk street")
```

USER what is this church?

```
dialogAct(set_question)
*isA(id:X2, type:church)
index(id:X2)
```

WIZARD sorry, can you say this again?

```
dialogAct(repeat)
```

USER i said what is this church on my left!

```
dialogAct(set_question)
*isA(id:X2, type:church)
index(id:X2)
position(id:X2, ref:@USER,
         location:left)
```

WIZARD it is saint john's.

```
dialogAct(inform)
isA(id:X3, type:church)
*isNamed(id:X3, name:"saint john's")
equivalent(id:X2, id:X3)
```

USER A sign here says it is saint mark's.

```
dialogAct(inform)
isA(id:X4, type:church)
*isNamed(id:X4, name:"saint mark's")
equivalent(id:X2, id:X4)
```

Figure 1: Sample annotated dialog

Annotation with the MRL is not text bound; i.e. we do not specify the alignment between tokens in the utterance and elements in the MR, apart from the verbatim string extracts. Even though some supervision can be helpful for automatically inferring such alignments [5], the annotation task is non-trivial, and often it is not possible to decide on a single correct alignment. For example, in the first utterance in Figure 1 the `isA` predicate denoting the restaurant cannot be aligned with a particular token or span in the utterance since neither "italian" nor "meal" on their own can represent it. Furthermore, including the tokens between "italian" and "meal" would not be ideal either since many of them are not relevant to this predicate.

2.1 Predicates

The MRL contains predicates with a variety of arguments to introduce entities, properties of entities, and their relations:

- Predicates introducing entities and their properties: `isA`, `isNamed` and `hasProperty`.
- Predicates describing user actions, such as `walk` and `turn`, with arguments such as `direction` and `along_location` to express the various modes of action.
- Predicates describing geographic relations, such as `distance`, `route` and `position`. The latter uses the argument `ref` in order to denote relative positioning.
- Predicates denoting whether an entity is introduced using a definite article (`def`), an indefinite (`indef`) or an indexical (`index`), which are useful in determining which real-world entity is being referred to.
- Predicates expressing numerical relations such as `argmin` and `argmax`.

3 Semantic parsing for the new corpus

The annotation format shown in Figure 1 is readable and easy to use, which allowed us to develop the annotation scheme and achieve high inter-annotator agreement (Deliverable 4.1.2). However, it is not straightforward to use in a structured prediction framework, nor in experiments, as it is difficult to compare different MRL expressions in this format beyond exact match. The problem with exact match is that it does not allow partial credit to be given to MRL expressions missing only a few predicates, and it considers all incorrect MRLs to be equally bad.

For these reasons, we converted the annotation scheme based on predicates into a form consisting of *conceptNodes* and arguments. Under this conversion, all predicates introducing entities (`isA`) and most predicates introducing relations among entities (e.g. `distance`) become conceptNodes, while all other predicates (e.g. `isNamed`, `def`, `hasProperty`) are converted into arguments of conceptNodes. For example, the MRL expression for the first utterance in Figure 1 is converted into the form shown in Figure 2g. This conversion resulted in 16 utterance-level labels (15 dialog acts plus an additional one for the non-interpretable utterances), 35 conceptNode types and 32 argument types. In case an entity is annotated in an utterance without explicit mention of its type (e.g. X2 in the last utterance of Figure 1), we denote it with a conceptNode with the special type `empty`. Each conceptNode has an identifier attached to it (e.g. X1) and each argument can take as value a constant (e.g. `det`), a conceptNode identifier (e.g. `location`) or a verbatim string extract from the utterance (e.g. `cuisine`). Any argument not present for a conceptNode in the utterance (e.g. the `name` of `restaurant`) is implicitly set to null.

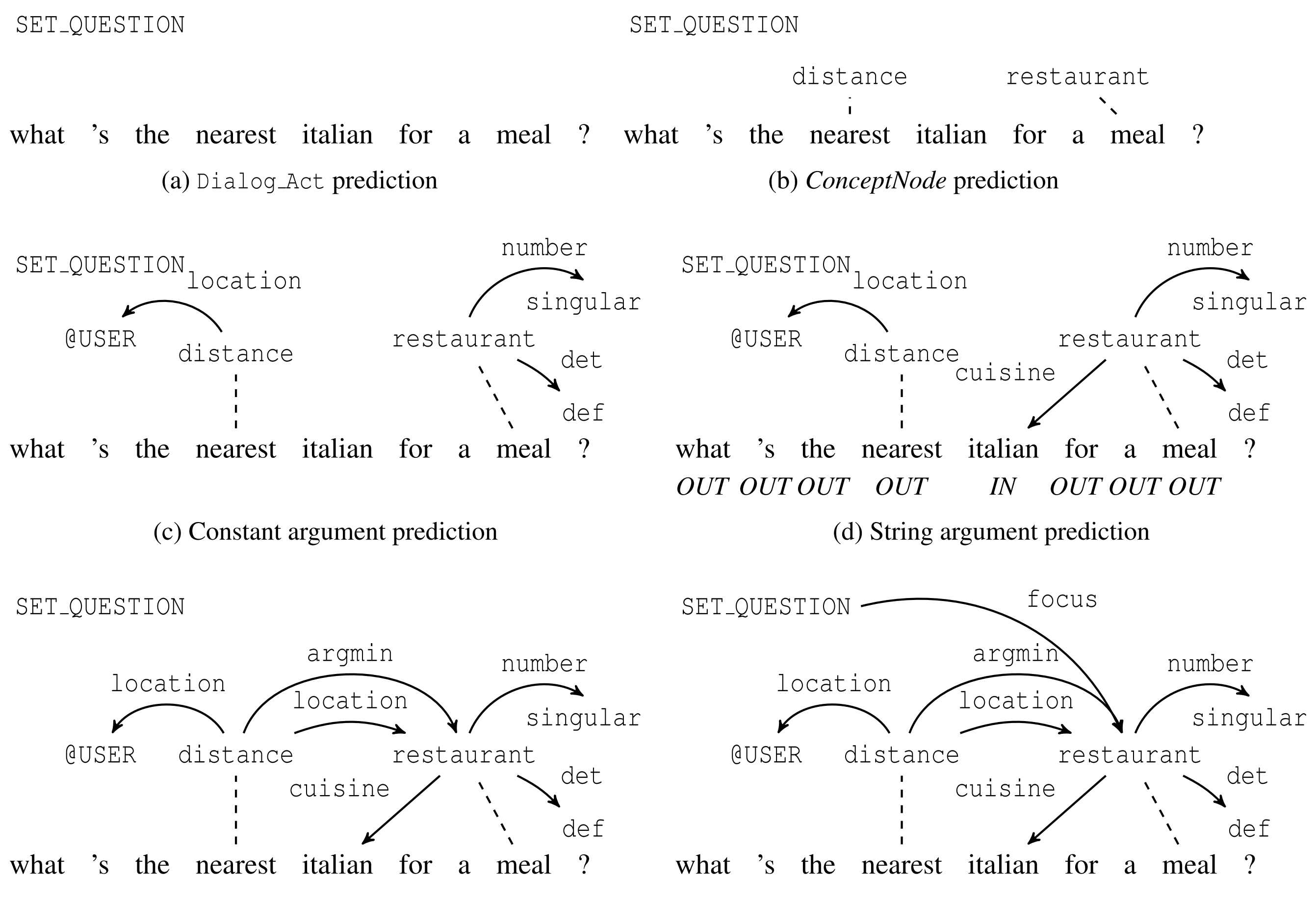
The comparison between a predicted conceptNode form and a gold standard one is needed for two reasons: for evaluation at test time, and during training. We perform the comparison in three stages. First we identify a mapping of the conceptNode identifiers to the ones of the gold standard. While the identifiers do not carry any semantics, they are necessary so that we can differentiate between multiple conceptNodes of the same type, e.g. if a second `restaurant` had been predicted in Figure 2h then it would need to have a different identifier and so it wouldn't be matched with any conceptNode in the gold standard. Second, we decompose the conceptNode forms for both into a set of predictions, as shown in Figure 2h. This decomposition allows us to reward a system for predicting parts of the MRL correctly, e.g. for predicting a `restaurant` in Figure 2h even if some of its arguments are incorrect. However, to receive credit for an argument, the conceptNode to which it belongs must be correct too. Using these predictions we count the true positives, false positives and false negatives, and calculate precision, recall and F-score. For a given pair of conceptNode forms, many mappings are possible. We choose one mapping by calculating all mappings allowed by the types of the conceptNodes (only mappings between identifiers of conceptNodes of the same type are allowed), and then selecting the one resulting in the lowest sum of false positives and false negatives.

4 Task decomposition

Given an utterance, the semantic parsing task as now defined is to predict the correct conceptNode form for it. We decompose this task in stages, as depicted in Figure 2 and described below.

Dialog act tagging In dialog act prediction, the task is to assign an utterance level label. For this purpose we build a classifier using features based on the textual content of the utterance and on the utterance preceding it. The former contain all unigrams, bigrams and trigrams and final punctuation mark (if any). Unlike in typical text classification tasks, content words are not always helpful in dialog act tagging; e.g. the word "church" in the user utterances in Figure 1 is not indicative of `set_question`, while n-grams of words typically considered as stopwords, such as "what is this", can be more helpful. The punctuation feature is used as a proxy to the prosodical information commonly used in dialog act tagging for spoken language [6]. The features based on the preceding utterance contain whether it was by the user or the wizard and in the latter case its dialog act. Such features are useful in determining the act of short, ambiguous utterances such as "yes" which is tagged as `yes`, when following a `prop_question` utterance by the wizard but as `acknowledge` otherwise. If the dialog act predicted is to be accompanied by other predicates then we proceed to the remaining stages, otherwise the prediction is terminated.

ConceptNode prediction In conceptNode prediction the task is to predict whether each of the tokens in the utterance denotes a conceptNode of a particular type, or null, in a left-to-right fashion, thus forming a 36-class tagging task. For example, in Figure 2g the conceptNode `distance` is predicted from the token "nearest" and the conceptNode `restaurant` from the token "meal", while all other tokens predict null (see Figure 2b). The features used include the target token and its lemma, which are conjoined with the part-of-speech tag, the previous and following tokens, as well as the the lemmas of the tokens with which it has syntactic dependencies. Further features represent the dialog act predicted (e.g. `route` is more likely be present in an utterance tagged as `set_question`), the types of the conceptNodes already



(a) Dialog_Act prediction

(b) *ConceptNode* prediction

(c) Constant argument prediction

(d) String argument prediction

(e) Node argument prediction

(f) Focus/Negation prediction

```
SET_QUESTION(focus:X1)
X1:restaurant(num:singular, det:def,
    cuisine:“italian”)
X2:distance(location:@USER, location:X1,
    argmin:X1)
```

(g) *ConceptNode* form

```
dialogAct:SET_QUESTION
X1:restaurant
X1:restaurant(num:singular)
X1:restaurant(det:def)
X1:restaurant(cuisine:“italian”)
X2:distance
X2:distance(location:@USER)
X2:distance(location:X1-restaurant
                    (num:singular, det:def))
X2:distance(argmin:X1)
focus:X1-restaurant(num:singular)
```

(h) Evaluation form

Figure 2: The stages followed by the semantic parser during prediction.

predicted in the utterance, as well as the total number of conceptNodes predicted so far. To repeat an important comment made earlier, note that the alignments between conceptNodes and tokens are not part of the manual annotation, and so from a machine learning perspective are considered *hidden*. Given that the evaluation does not consider the alignment between conceptNodes and tokens, it would have been equally correct to have predicted the same conceptNodes from any other token; e.g. `restaurant` could have been predicted from "italian". Nevertheless, the choice of alignment is likely to affect the argument prediction stages that follow, since it determines feature extraction for these stages. Hence it is important that the semantic parser learns to align well.

Constant argument prediction Each of the conceptNodes predicted in the previous stage has arguments that need to be filled in. In Figure 2c, conceptNodes denoting entities such as `restaurant` have the arguments `name`, `number`, `det`, `cuisine`, etc., while `distance` has two `location` arguments and one `argmin`. In this stage we decide, for each predicate's argument, whether its value should be one of the constants defined (e.g. the values `@USER`, `singular` and `def` for argument types `location`, `number` and `det`), or the constant `STRING` to indicate that the value should be a verbatim extract from the utterance (e.g. argument `cuisine`) or `NODE` to indicate that the value should be a node (e.g. the arguments `location` and `argmin`). If an argument is not present (e.g. `name` for the `restaurant`), it is set to the special constant `NULL`. Thus this stage is a multiclass classification task in which the correct constant must be predicted for each argument. Since each argument type has its own set of constants, we use a different classifier for each of them, resulting in 32 classifiers for this stage that are applied according to the conceptNodes predicted.

The features used for each classifier consist of the conceptNode type (different entities are likely to have different arguments filled in, e.g. `size` is more likely to be filled in for `church` rather than a `restaurant`), the token as well as its lemma and part-of-speech tag that predicted the conceptNode (e.g. the `number` argument can be easily predicted from these), and the syntactic dependency paths from the token that predicted the conceptNode (e.g. "nearest" for `distance`) to all other tokens in the utterance. Furthermore, we include as features the values already predicted for the other arguments (e.g. the `name` argument is more likely to be null if the `number` is predicted to be `plural`), the dialog act and the other conceptNode types predicted in the utterance.

String argument prediction For each argument with `STRING` predicted as its value (e.g. `cuisine` in Figure 2d), we predict for each token in left-to-right order whether it should be part of the verbatim string extract for this argument or not (*IN* or *OUT*). Since the string extracts that are appropriate for each argument differ (e.g. the values for `cuisine` are unlikely to be the same as those for `name`), we use separate binary classifiers for each of them. The features used include the target token and its lemma, which are conjoined with the part-of-speech tag, the previous and following tokens, as well as the lemmas of the tokens with which it has syntactic dependencies. Further features include the label assigned to the previous token and the syntactic dependency path from the token that predicted the conceptNode to the target token.

Node argument prediction For each argument with `NODE` predicted as its value, we predict for each conceptNode whether it should be the value for it, e.g. in Figure 2e the `restaurant` conceptNode is the value for both `location` and `argmin` arguments of `distance`. As with the string argument prediction, we use separate binary classifiers for each argument. The features extracted include the token and the lemma and part-of-speech tag of the token that predicted the argument conceptNode (e.g. "meal" for `restaurant`), as well as the syntactic dependency path between the token that predicted the argument conceptNode and the token that predicted the conceptNode of the feature in question.

Focus/Negation prediction We treat the prediction of which conceptNode(s) should be focused or negated as two separate binary classification tasks. The features used for this include the token that predicted the conceptNode in question, its lemma and PoS tags and the syntactic dependency paths to all other tokens in the utterance. Further features include the type of the conceptNode in question and the features predicted for it.

5 Imitation learning for structured prediction

In order to learn the classifiers for the task decomposition described above we need to address two challenges. The first challenge is the complexity of the structure to be predicted. The task involves many interdependent predictions from a variety of classifiers; thus it doesn't lend itself to modelling approaches that rely on predicting a particular type of graph. In addition, incorporating features that capture these dependencies is likely to result in increased predictive accuracy. The second challenge is the lack of alignment information during training. As discussed earlier, aligning predicates with tokens from the utterances is a non-trivial annotation task, and it is ignored by the evaluation since it is not part of the MR expression. Due to the lack of this information we are unable to train a pipeline of independent classifiers; for example in order to train a classifier for conceptNode prediction, we would need to know which token is aligned with `restaurant` in Figure 2g. Furthermore, many of the features extracted for all stages beyond the dialog act tagging depend on the alignment between the tokens and the predicted conceptNodes, i.e. rely on predictions made earlier that break the commonly assumed first- or second-order Markov assumption.
Imitation learning algorithms aim at learning controllers from demonstrations by human experts [7, 8]. Unlike standard reinforcement learning algorithms [9], they do not require the specification of a reward function by the user. Instead, the algorithm observes a human expert performing a sequence of actions to predict task instances and learns a policy that "imitates" the expert with the purpose of generalizing to unseen data. Imitation learning algorithms such as SEARN [10] and DAGGER [11] have been applied successfully to a variety of structured prediction tasks such as summarization, biomedical event extraction and dynamic feature selection [10, 12] thanks to their flexibility in incorporating features based on structure. In this work we focus on DAGGER and extend it to handle the missing alignments in the training data.

5.1 DAgger

The dataset aggregation (DAGGER) algorithm [11] forms the prediction of an instance s as a sequence of T actions $\hat{y}_{1:T}$ predicted by a learned policy which consists of one or more classifiers. During training, DAGGER converts the problem of learning how to predict these sequences of actions into cost sensitive classification (CSC) learning. In CSC each training example has a vector of misclassification costs associated with it, thus rendering some mistakes on some examples to be more expensive than others [13]. The dependencies between the actions are learnt by appropriate generation of CSC examples. In particular, the features for each action y_t can take into account all previous actions $\hat{y}_{1:t-1}$, while the costs for each possible action c_t take into account the effect of y_t on the remaining sequence of actions. The costs for each possible action y_t^j are estimated by assuming that the action y_t^j was taken, then the following actions

Algorithm 1: Imitation learning with DAGGER

```
Input: training instances $\mathcal{S}$, expert policy $\pi^\star$,
loss function $\ell$, learning rate $\beta$, CSC learner $CSCL$
Output: Learned policy $H_N$
1  CSC Examples $E = \emptyset$
2  for $i = 1$ to $N$ do
3      $p = (1-\beta)^{i-1}$
4      current policy $\pi = p\pi^\star + (1-p)H_{i-1}$
5      for $s$ in $S$ do
6          Predict $\pi(s) = \hat{y}_{1:T}$
7          for $\hat{y}_t$ in $\pi(s)$ do
8              Extract features $\Phi_t = f(s, \hat{y}_{1:t-1})$
9              foreach possible action $y_t^j$ do
10                 Predict $y\prime_{t+1:T} = \pi(s; \hat{y}_{1:t-1}, y_t^j)$
11                 Assess $c_t^j = \ell(\hat{y}_{1:t-1}, y_t^j, y\prime_{t+1:T})$
12             Add $(\Phi_t, c_t)$ to $E$
13     Learn $H_i = CSCL(E)$
```

for that instance $y_{t+1:T}$ are predicted using the learned policy H_i, and finally the whole sequence of actions is compared against the correct output for that instance.

Algorithm 1 presents the training procedure of DAGGER in more detail. DAGGER requires a set of labeled training instances $\mathcal{S}$ and a loss function ℓ that compares complete action sequences for instances in $\mathcal{S}$ against the correct output for them. In addition, an *expert policy* $\pi^\star$ must be specified which is an oracle that returns the optimal action $\hat{y}_t$ for the instances in the training data, which is akin to an expert demonstrating the task. $\pi^\star$ is typically derived from the labels of the training instances; for example in part of speech tagging $\pi^\star$ would return the correct tag for each token. In addition, the learning rate β and a CSC learner (*CSCL*) must be provided. The algorithm outputs a learned policy H_N that, unlike the expert policy $\pi^\star$, can generalize to unseen data.

Each training iteration begins by setting the probability p (line 3) of using $\pi^\star$ in the current policy π. In the first iteration only $\pi^\star$ is used but in later iterations π becomes stochastic as for each action we use $\pi^\star$ with probability p and the learned policy from the previous iteration h_{i-1} with probability $1-p$ (line 4). Then π is used to predict each training instance s (line 6). For each action $\hat{y}_t$, a CSC example is generated (lines 7-12). The features Φ_t are extracted from s and the previous actions $\hat{y}_{1:t-1}$ (line 8) and are expected to be useful in predicting the current action $\hat{y}_t$. The cost for each possible action y_t^j is estimated by predicting the remaining actions $y\prime_{t+1:T}$ needed for s using π (line 10) and calculating the loss incurred given y_t^j w.r.t. the correct output for s using ℓ (line 11). As π is stochastic, it is common to use multiple samples to assess the cost of each action by repeating these steps. The features together with the costs for each possible action form a CSC example (Φ_t, c_t) (line 12). At the end of each iteration the CSC examples obtained from all iterations are used by the CSC learning algorithm to learn the classifiers for a new H_i (line 13).

When predicting the training instances (line 6) and when estimating the costs for each action (lines 10-11), the policy learned in the previous iteration H_{i-1} is used since it is part of the current policy π, thus the CSC examples used to learn H_i in the current iteration depend on the predictions of H_{i-1}. The degree to which it is used depends on the probability p set at the beginning of each iteration. By gradually decreasing the use of the expert policy in the current policy, the learned policy is adjusted to its own predictions, thus learning how to predict sequences of actions jointly. The learning rate β determines how fast π moves away from $\pi^\star$.

5.2 Handling missing labels

As mentioned in the previous section, when generating a CSC training example in DAGGER (lines 7-12), we do not need to know whether an action y_t^j is correct or not, we only evaluate what the effect of y_t^j is on the loss incurred by the complete action sequence. In other words, the loss function does not need to decompose over the actions in the sequence in order to evaluate them independently. The classifiers forming H_i are trained to predict actions that minimize the loss on the instance, independently of what the correct action might have been.

Non-decomposable loss functions can be very useful when the training data has missing labels, as is the case in our semantic parsing task. For the semantic parsing task, the loss function is the sum of the false positive and false negative predictions used in calculating the precision and recall defined in Section 3. This evaluation ignores the alignment between tokens and conceptNodes, since this information is not part of the annotation, and only considers whether the correct conceptNodes were generated. Thus it does not decompose over the actions used to predict the output for an utterance. While this prohibits us from training a classifier for the conceptNode prediction stage in a straightforward manner, as we do not have direct supervision, we can still train one using DAGGER . As explained in the previous section, the classifier for this stage will be learnt so that it predicts in a way that minimizes the loss according to the evaluation used. For example, while it could infer that the conceptNode `restaurant` is predicted by the token “what”, it is more likely to infer that it is predicted by the token “italian” since the latter would facilitate the correct prediction of other parts of the output, e.g. that the conceptNode predicted is one of the `location` arguments of the `distance` conceptNode.

The only component in DAgger for which knowledge of the correct actions is assumed is the expert policy $\pi^\star$ for the training instances. As discussed earlier, due to the lack of alignment information in our annotation the correct actions are not fully specified, i.e. at the conceptNode prediction stage we do not know which tokens predict which conceptNodes, only the conceptNodes that need to be predicted for the utterance. In order to overcome this issue, we develop a randomised expert policy in which actions that cannot specified by the annotation are chosen randomly. For example, in Figure 2b when considering which action to predict for token “what” (choice between null, `distance`, or `restaurant`), the expert policy picks among them at random so that by the end of the predictions for this stage the correct conceptNodes have been predicted. As a result, the expert policy needs to be dynamic, since the choice of conceptNode type for a particular token depends on the ones already predicted.

The actions returned by such an expert policy are likely to be sub-optimal; however, since the expert policy is progressively replaced by the learned hypothesis in the current policy, its effect on the training process is decreased. For example, in the first iteration the CSC examples generated for the conceptNode prediction stage in Figure 2b will teach the classifier that any of the tokens can predict the conceptNodes needed, since, in assessing the cost of each action, the actions predicted by the current policy (line 10) are all returned by the expert policy which dynamically adjusts so that it minimizes the loss w.r.t the gold standard. In subsequent iterations, though, the actions predicted by the current policy are returned with increasing frequency by the learned hypothesis; hence the loss incurred by sub-optimal actions (e.g. predicting `distance` from token “what”) is likely to be higher than that of more reasonable ones (e.g.

	basic			+align		
	Rec	Prec	F	Rec	Prec	F
dialog act	77%			80.5		
conceptNodes	68.7	45.7	54.8	75.5	51.7	61.4
arguments	73.9	73.7	73.8	76.8	77.3	77.1
focus	87.1	80.7	83.8	86	81.2	83.6
overall	56.6	42.3	48.4	63.5	46.2	53.5
exact match	47.9%			49.1%		

Table 1: Performances on the test data. Dialog act tagging and exact match measured with accuracy.

predicting it from the token "nearest") since the classifiers learned for the argument extraction stages are more likely to make correct predictions. While being able to learn without alignment information between conceptNodes and tokens in the training utterances is desirable, it would still be useful to incorporate some easy-to-obtain supervision for this stage, e.g. that token "street" predicts the conceptNode `street`, or that "walking" predicts `walk`. Such alignment supervision is easy to incorporate in the training process proposed by augmenting the expert policy with a dictionary mapping tokens to conceptNode types. More specifically, when the action for a token is requested by the expert policy, the latter first checks the dictionary and, if the token is mapped to a conceptNode type, and if such a type is still needed to correctly predict the MR for the utterance, then a conceptNode of that type is returned. If the token is not in the dictionary, then the randomised method is used. Note that this alignment information is not used during testing, and the training process is not constrained by it since the learning algorithm might eventually learn different alignments as the expert policy is progressively ignored.

6 Experiments

For the cost-sensitive classification learning we used the adaptive regularization of weight vectors (AROW) learning algorithm [14]. AROW is an online learning algorithm for linear predictors that takes into account the rarity of each feature and adjusts the per-feature learning rates so that popular features do not overshadow rare but useful ones. Following [15], we also implemented *focused costing* for the three argument prediction stages by assuming that the prediction of a particular argument does not affect the prediction of the remaining ones. Finally, we restricted the prediction of conceptNodes to content word tokens, since prepositions, articles, etc. are less likely to result in useful alignments.

In order to assess the performance of the semantic parser, we split the annotated dialogs into two sets, one for training (and development) and one for testing. The former consists of four dialogs of the first scenario and seven from the second (11 in total), while the latter consists of three dialogs from each scenario (six in total). All development and feature engineering was conducted using cross-validation at the dialog level on the training set. Cross-validation at the dialog level instead of the utterance level ensures that each fold contains utterances from all parts of the scenario from which the dialog is taken. Using this setup, we set the parameters for DAGGER to 12 training iterations, with the learning rate β set to 0.3.

The results reported in Table 1 were obtained by training on dialogs from the training set and evaluating on dialogs from the test set. The overall performance of our approach (column "basic") in terms of predicting the elements of the conceptNode form (Figure 2g) is 48.4 points in F-score. The exact match accuracy at the utterance level is 47.9%. Isolating the performance at the conceptNodes and the argument prediction stages, we observe that the main bottleneck is the former, which is 54.8 points in F-score compared to 73.8 for the latter.

In order to improve conceptNode prediction performance we augmented the expert policy with an alignment dictionary extracted from the training data, containing tokens that commonly predict a particular conceptNode type, as described in the previous section. These included nouns (e.g. "cash"-`atm`), verbs ("see"-`isVisible`) and tokens of names (e.g. "KFC"-`restaurant`). In total 95 tokens were collected. As shown in Table 1 (column "+align") the incorporation of the alignment dictionary improved not only conceptNode prediction performance by 6.6 points in F-score, but also argument prediction by 3.3 points, thus demonstrating the benefits of joint learning. The overall prediction performance improved by 4.9 points in F-score, and whole exact match accuracy improved by 1.2 percentage points.

In the experimental setup considered above, all training dialogs from both scenarios were used to train the taggers, which were then evaluated on all testing dialogs from both scenarios. While this is a reasonable evaluation approach, it is likely to be relatively forgiving, since in practice semantic parsers are likely to encounter scenarios unseen in their training/development data. Similar setups are commonly used in semantic parsing evaluations on the ATIS corpus, in which dialogs concerning the same flight request are included in training, development and testing sets. Therefore we considered a second evaluation setup in which the dialogs used to train the taggers are from different scenario(s) than the scenario of the dialogs used to evaluate their performance. Testing dialogs from different scenarios is likely to be more challenging since they are likely to contain mentions to different entities, thus they provide a stricter evaluation of the generalization performance. When testing on the dialogs from the first scenario and training on the dialogs from the the second one the overall performance using the alignment dictionary was 36.9 points in F-score, while in the reverse direction it was 41.7. While direct comparisons against the performances reported in Table 1 would not be meaningful since fewer dialogs are being used for both training and testing, we consider evaluating semantic parsers in this way provides useful insights into their likely performance in a real-world setting.

7 Discussion

Previous work on learning semantic parsers has handled the issue of aligning tokens with predicates in a variety of ways. [16] manually engineered a CCG lexicon for their experiments with the ATIS corpus. [17] used a dedicated algorithm to infer a similar dictionary as part of the training process, and used alignments obtained from the standard machine tranlsation alignment tool Giza++ in order to initialize some of the features. Most recent work on the GeoQuery dataset uses an alignment dictionary that includes for each geographical entity all noun phrases used to refer to it, for example "usa", "us", "country"-`usa` [18]. Applying such algorithms as part of semantic parsing learning is reminiscent of the use of alignment approaches as part of machine translation learning in order to restrict the search space.

In contrast, our approach infers the alignments jointly with learning the semantic parser so that the alignments inferred facilitate the correct MR prediction. While we were able to improve performance using an alignment dictionary, performance was not drastically worse without it.

The performance on the new corpus is lower than those commonly reported on ATIS and GeoQuery. This is primarily due to the wider controlled vocabulary, which is indicative of its wider domain (see

Deliverable 4.1.2). Although the new corpus has fewer utterances than ATIS, it has a wider vocabulary of NL words and the utterances themselves are more varied since they do not consist of database queries exclusively.

The main performance bottleneck is the sparsity of the training data. Some of the rarer predicates appear only a few times in relatively long utterances; thus it is difficult to learn appropriate alignments for them. Unlike machine translation between natural languages, in which we can exploit large parallel corpora, it is unrealistic to expect large quantities of utterances to be annotated with MR expressions. An appealing solution would be to use response-based learning, i.e. use the response from the system instead of MR expressions as training signal [19]. However such an approach would not be straightforward to implement in context-dependent semantic parsing since the response from the system would affect the following utterances, thus requiring the development of user simulators, a non-trivial task [20] which is beyond the scope of this work.

Finally, dialog context is taken into account in predicting the dialog act tag for each utterance. Even though the dialogs are annotated with coreference links we did not attempt to perform this task as it is difficult to evaluate and the performance on conceptNode prediction on which it relies it is relatively low. Nevertheless, we anticipate coreference resolution in the new corpus to be more difficult than in ATIS since the dialogs are 15 times as long as the the dialogs of that corpus.

8 Conclusions, Difficulties and Future Work

We have developed a semantic parser for the SPACEBOOK corpus under the imitation learning paradigm. In particular, we extended the algorithm DAGGER to handle the absence of alignment information from the training data by developing a randomized dynamic expert policy. We improved its performance by incorporating an alignment dictionary, reaching 53.5 points in overall F-score performance.

Evaluation was intrinsic, testing how well the semantic parser is able to predict the MRLs for unseen sentences in a held-out portion of the corpus. Our original intention was to integrate the machine-learned semantic parser into the full SPACEBOOK system, and see how well the system performed with this parser compared to the rule-based parser which is currently being used. However, the semantic parser was not developed in time for such integration, only being completed with a few months of the project remaining. The main reason that the semantic parser was not developed in time was the delay in obtaining the Wizard-of-Oz data at the start of the project, from which we were never able to recover, especially since creation of a manually annotated corpus is a significant, time-consuming exercise, requiring many months of annotation time (as described in Deliverable 4.1.2).

Obvious areas for future work include investigating the use of continuous semantic vector representations in order to ameliorate the data sparsity issues, and exploiting the coreference information in the semantic parsing corpus to learn a coreference resolution system, which will make the semantic parser more contextually aware and lead to even richer semantic output.

References

[1] Mitchell P. Marcus, Mary A. Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of English: the Penn treebank. *Computational Linguistics*, 19(2):313–330, June 1993.

[2] Deborah A. Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. Expanding the scope of the ATIS task: the ATIS-3 corpus. In *Proceedings of the Workshop on Human Language Technology*, pages 43–48, Plainsboro, New Jersey, 1994.

[3] John M. Zelle. *Using Inductive Logic Programming to Automate the Construction of Natural Language Parsers*. PhD thesis, Department of Computer Sciences, The University of Texas at Austin, 1995.

[4] Ann Copestake, Dan Flickinger, Ivan Sag, and Carl Pollard. Minimal recursion semantics: An introduction. *Research in Language and Computation*, 3(2–3):281–332, 2005.

[5] Percy Liang, Michael I. Jordan, and Dan Klein. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 91–99, Singapore, 2009.

[6] Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Rachel Martin, Carol Van Ess-Dykema, and Marie Meteer. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational linguistics*, 26(3):339–373, 2000.

[7] Pieter Abbeel. *Apprenticeship Learning and Reinforcement Learning with Application to Robotic Control*. PhD thesis, Department of Computer Science, Stanford University, 2008.

[8] Umar Syed. *Reinforcement learning without rewards*. PhD thesis, Department of Computer Science, Princeton University, 2010.

[9] Richard Sutton and Andrew Barto. *Reinforcement learning: An introduction*. MIT press, 1996.

[10] Hal Daumé III, John Langford, and Daniel Marcu. Search-based structured prediction. *Machine Learning*, 75:297–325, 2009.

[11] Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *14th International Conference on Artificial Intelligence and Statistics*, pages 627–635, 2011.

[12] Andreas Vlachos. An investigation of imitation learning algorithms for structured prediction. *Journal of Machine Learning Research Workshop and Conference Proceedings, Proceedings of the 10th European Workshop on Reinforcement Learning*, 24:143–154, 2012.

[13] Pedro Domingos. Metacost: a general method for making classifiers cost-sensitive. In *Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining*, pages 155–164. Association for Computing Machinery, 1999.

[14] Koby Crammer, Alex Kulesza, and Mark Dredze. Adaptive regularization of weight vectors. In *Advances in Neural Information Processing Systems 22*, pages 414–422, 2009.

[15] Andreas Vlachos and Mark Craven. Search-based structured prediction applied to biomedical event extraction. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 49–57. Association for Computational Linguistics, 2011.

[16] Luke S. Zettlemoyer and Michael Collins. Learning context-dependent mappings from sentences to logical form. In *Proceedings of the Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pages 976–984, Singapore, 2009.

[17] Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. Lexical generalization in CCG grammar induction for semantic parsing. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1512–1523, Edinburgh, UK, 2011.

[18] Bevan Keeley Jones, Mark Johnson, and Sharon Goldwater. Semantic parsing with Bayesian tree transducers. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 488–496, 2012.

[19] Percy Liang, Michael Jordan, and Dan Klein. Learning dependency-based compositional semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 590–599, Portland, Oregon, 2011.

[20] Simon Keizer, Stphane Rossignol, Senthilkumar Chandramohan, and Olivier Pietquin. User simulation in the development of statistical spoken dialogue systems. In Oliver Lemon and Olivier Pietquin, editors, *Data-Driven Methods for Adaptive Spoken Dialogue Systems*, pages 39–73. Springer New York, 2012.