Imitation learning for language generation from unaligned data

Gerasimos Lampouras
Department of Computer Science
University of Sheffield, UK
g.lampouras@sheffield.ac.uk

Andreas Vlachos
Department of Computer Science
University of Sheffield, UK
a.vlachos@sheffield.ac.uk

Abstract

Natural language generation (NLG) is the task of generating natural language from a meaning representation. Rule-based approaches require domain-specific and manually constructed linguistic resources, while most corpus based approaches rely on aligned training data and/or phrase templates. The latter are needed to restrict the search space for the structured prediction task defined by the unaligned datasets. In this work we propose the use of imitation learning for structured prediction which learns an incremental model that handles the large search space while avoiding explicitly enumerating it. We adapted the Locally Optimal Learning to Search (Chang et al., 2015) framework which allows us to train against non-decomposable loss functions such as the BLEU or ROUGE scores while not assuming gold standard alignments. We evaluate our approach on three datasets using both automatic measures and human judgements and achieve results comparable to the state-of-the-art approaches developed for each of them. Furthermore, we performed an analysis of the datasets which examines common issues with NLG evaluation.

1 Introduction

Natural language generation (NLG) is the task of generating natural language from a machine-interpretable meaning representation (MR). Traditionally, NLG systems tend to be rule-based and require domain-specific language resources (i.e. sentence plans, aggregation rules, ordering information, etc.) to be manually authored for a particular task, e.g. weather reporting (Reiter et al., 2005), route navigation (Dale et al., 2003), or ontology descriptions (Bontcheva and Wilks, 2004; Androutsopoulos et al., 2013).

From a machine learning perspective (ML) perspective, NLG is a complex structured prediction task due to its large output space, which is the set of all possible NL utterances. To limit the output space, many ML-based approaches rely on aligned training datasets (Mairesse et al., 2010; Dethlefs et al., 2013) that specify the words that each MR element is responsible for, and phrase templates. While these enable a variety of models to be trained, their applicability is restricted as they need additional manual annotation.

More recently, ML-based approaches focused on learning NLG models from unaligned training data. Dušek and Jurcícek (2015) learn how to incrementally generate deep-syntax dependency trees of candidate sentence plans specifying which MR elements to be mentioned and the overall sentence structure, however the surface realization (i.e. the actual NL word generation) is performed using rules based on the frames of an English-Czech dependency treebank, a resource that may not be available for certain domains or languages. In another approach, Wen et al. (2015) use a Long Short-term Memory (LSTM) network to learn from unaligned data and jointly address sentence planning and surface realization. They augment each cell of the LSTM with a gate that conditions it on the input MR. However the LSTM is trained using cross-entropy at the word level, thus unable to take into account interactions among the word being considered and words to be predicted later in the sentence, an issue which was ameliorated by training a backward LSTM to re-rank the best-scoring outputs generated by the forward one.

In this paper, we propose an ML-based approach that learns sentence planning and surface realization from unaligned training data using the imitation learning Locally Optimal Learning to Search (LOLS)

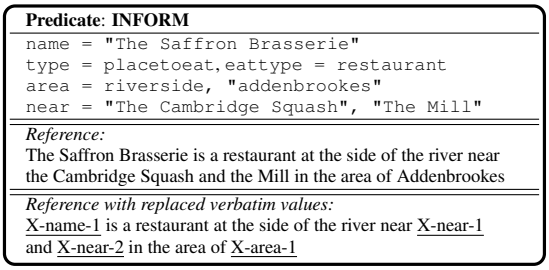

Figure 1: Sample MR and corresponding NL utterance from the BAGEL dataset.

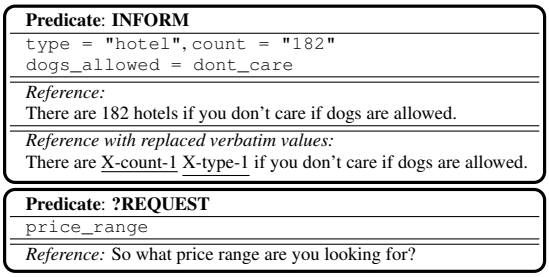

Figure 2: Sample MRs and corresponding NL utterances from the SF datasets.

framework (Chang et al., 2015). Similar to other imitation learning algorithms for structured prediction such as DAGGER (Ross et al., 2011), LOLS reduces structured prediction to classification with greedy inference thus avoiding the enumeration of all outputs, while also ameliorating the issue of error propagation. Unlike other structured prediction frameworks, LOLS is able to learn using non-decomposable loss functions. Thus it is well-suited to learning NLG systems where measures such as BLEU (Papineni et al., 2002) or ROUGE (Lin, 2004) that do not decompose over single words are commonly used for evaluation. Thus we are able to learn the interactions of the word currently being predicted with those to be predicted later in the sentence. We further propose a variant of LOLS using sequence correction and updates (Collins and Roark, 2004), and an exponential decay schedule (Daumé III et al., 2009) . We compare against the systems by Dušek and Jurcícek (2015) and Wen et al. (2015) on their respective datasets (three datasets across two domains) and show that the NLG system proposed achieves comparable results in both automatic and human evaluations.[1]

2 Natural language generation

The NLG process takes as input a meaning representation (MR) consisting of a predicate followed by an unordered set of attributes and corresponding values; the output is a NL sentence. Figures 1 and 2 show samples from the BAGEL (Mairesse et al., 2010) and SF datasets (Wen et al., 2015) that we are considering in this work. The predicates in each MR are utterance-level labels that represent the overall function of the utterance, e.g. whether the utterance should *inform* the user of the attributes and values, or whether it should *request* them from the user. Each attribute may have multiple values, which may be either strings that appear verbatim in the references (e.g. "the Saffron Brasserie", "hotel"), constants from a controlled vocabulary (e.g. `placetoeat`), or boolean (e.g. `yes`, `no`).

We preprocess the MR-NL pairs by replacing verbatim strings with variables ("X-") in the MRs and NLs, which results in the same delexicalized MR paired with multiple NL references (see Figures 1 and 2). Note that the second MR in Figure 2 contains no verbatim strings, and its reference is not modified. Multiple references for the same MR can be beneficial since they capture lexical variation for the same meaning, but also pose a challenge to model learning as they provide ambiguous training signal.

We formulate the generation of a NL utterance from a MR as a sequence of two types of actions, content prediction actions a_c and word prediction actions a_w (Alg. 1). To generate an NL utterance, each content prediction action selects which attribute c should be expressed next; in the case of multiple-valued attributes, one value is chosen based on their order of appearance in the training data. Once the content prediction action sequence is completed, for each selected attribute c, we generate a sequence of words chosen from its corresponding dictionary D_c. This dictionary consists of all the words that we have observed to co-occur with attribute c in the training data. Content and word prediction have special termination actions (END_{attr} and END_{word} respectively). The generation process can stop without exhausting all attributes in the MR thus making it possible to omit information deemed redundant (e.g. that a restaurant is a place to eat). From the action sequence produced, it is straightforward to derive the final sentence by keeping the word prediction actions; the content prediction actions are discarded.

[1] All code is available at `https://github.com/glampouras/JLOLS_NLG`

Algorithm 1: NLG process

Input: meaning representation MR with set of attributes C, attribute dictionaries $D_c, \forall c \in C$
Output: action sequence A

do
  predict attribute $c \in C \cup \{END_{attr}\}$ and append a_c to A_c
  remove c from C
while $a_c \neq END_{attr}$
for a_c *in* A_c **do**
  do
    predict word $w \in D_c \cup \{END_{word}\}$ and append a_w to A_w
  while $a_w \neq END_{word}$
$A = (A_c, A_w)$

Fig. 3 illustrates the action sequence to generate the NL sentence for the MR of Figure 1. The first set of actions is to choose amongst the attributes (`eattype`, `name`, `near`, or the content termination action END_{attr}), and if necessary also choose which value to express (e.g. for `area`, either `riverside` or `X-area-1`). Following this, for each chosen attribute and value we generate an appropriate sequence of word prediction actions (e.g. "at", "the", "side", "of", "the", "river") followed by the END_{word} action denoted by "|" (top part of Fig. 3). Content prediction actions are only used indirectly to generate the sentence; thus different sequences can result in the same sentence, as shown in the bottom part of Fig. 3.

The NLG process defined in this section assumes we train two types of classifiers, one for the content prediction actions and one for word prediction actions for each attribute. If we had alignment information, it would be possible to extract data to train both types of models, either independently (Angeli et al., 2010) or jointly. However, we do not assume access to such information. Instead, we take advantage of the ability of imitation learning algorithms such as LOLS to learn with non-decomposable loss functions by only needing to evaluate complete output predictions instead of individual actions. In NLG's case, this means we do not require explicit supervision for the content and word prediction actions, but only a way to evaluate complete generated sentences against the reference using measures such as BLEU.

3 Locally Optimal Learning to Search

Here we describe how we learn the content and word classifiers we introduced in Section 2. As input to the algorithm (Alg. 2), we assume a set of training instances $\mathcal{S}$ and a loss function ℓ that compares complete NL sequences generated for MRs in $\mathcal{S}$ against references for that MR, e.g. using BLEU or ROUGE. In addition, an expert policy π^{ref} must be specified which will be acting as an oracle during training, returning the best content or word action possible given the words predicted already and the gold standard NL reference for the instance. We describe the expert policy and loss function in detail in separate subsections. Finally, the learning rate β is also part of the input. The output is a learned policy consisting of one classifier for content prediction and one classifier for word prediction for each attribute c, whose label set is the attribute dictionary D_c. These classifiers are learned using a cost-sensitive classification (*CSC*) learning algorithm (*CSCL*). In *CSC* each training example has a vector of misclassification costs, thus rendering some mistakes on some examples more expensive than others.

Before the training iterations begin, a policy π_0 (i.e. content and word classifiers) is initialized on

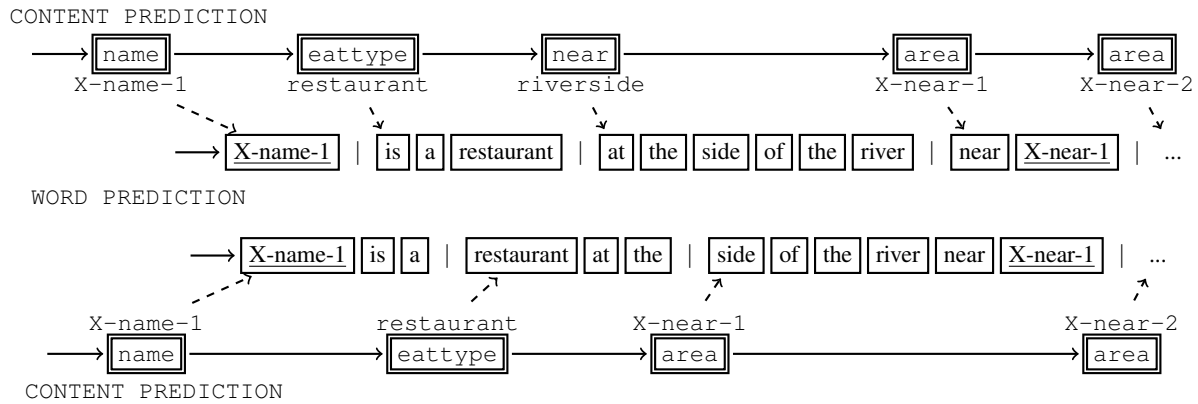

Figure 3: NLG process example with two different content and word prediction action sequences.

Algorithm 2: Locally Optimal Learning to Search (LOLS) for NLG

Input: training instances $\mathcal{S}$, expert policy π^{ref}, loss function ℓ, learning rate β, CSC learner *CSCL*
Output: Learned policy H_N

1 Initialize a policy π_0
2 **for** $i = 0$ **to** N **do**
3 CSC examples $E = \emptyset$
4 $p = (1-\beta)^i$ // exponential decay
5 **for** s ***in*** S **do**
6 Predict A by executing $\pi_i(s)$ // roll-in
7 **for** $action\ a_t$ ***in*** A **do**
8 Let $\pi^{out} = \pi^{ref}$ with probability p, otherwise $\pi^{out} = \pi_i$
9 **foreach** *possible action* a_t^j **do**
10 $a\prime_{t+1:T} = \pi^{out}(s; a_{1:t-1}, a_t^j)$ // roll-out
11 Assess $c_t^j = \ell(a_{1:t-1}, a_t^j, a\prime_{t+1:T})$
12 Create a feature vector $\Phi_t = f(s, a_{1:t-1})$
13 Add (Φ_t, c_t) to E
14 Set $a^\star = a_t^j$ with minimum c_t^j // find best possible action
15 **if** $a_t \neq a^\star$ **then**
16 evaluate only the next E actions $a_{t+1} \dots a_{t+E}$ **in** A
17 then replace A by executing π^{ref} for actions $a_{1:t+E}$
18 and $\pi_i(s)$ for actions $a_{t+E+1:T}$ // sequence correction
19 Learn $\pi_{i+1} = CSCL(\pi_i, E)$ // update
20 $H_N = avg(\pi_0, \dots, \pi_N)$

the training data. For every instance s of the training data S, the algorithm predicts an action sequence $\{a_0 \dots a_T\}$ (line 6, roll-in) using the learned policy π_i of the previous iteration (if it is the first iteration, this is the initial learned policy π_0), using the NLG process defined in Alg.1. At each timestep t of this sequence, the algorithm considers all actions a_t^j that the (content or word) classifier could take (line 9). In order to estimate the cost of each action a_t^j, the algorithm produces an action sequence $\left\{a_{1:t-1}, a_t^j, a\prime_{t+1:T}\right\}$ (line 10, roll-out) assuming action a_t^j was taken, and assesses it according to the loss function ℓ (line 11). Thus the algorithm learns whether to select an attribute or generate a word by taking the corresponding action and assessing its effect on the quality of the complete NL. A *CSC* training example is generated from these costs (line 12-13), which will be used to update the appropriate classification model in the learned policy. The features Φ_t are extracted from the input MR and all previous actions and the partial NL corresponding to them (line 12) thus capturing (possibly long-range) dependencies between actions. Finally, all the policies trained at the end of each iteration are averaged into a final policy (line 19). The learned policies consisting of classifiers, unlike the expert policy π^{ref} which is available only for the training instances, can generalise to unseen data.

When examining all possible actions, the algorithm generates the rest of the alternative sequence (roll-out) $\{a\prime_{t+1:T}\}$ according to either the expert policy π^{ref} or the learned policy π_i, depending on the probability p. In its original specification (Algorithm 1 by Chang et al. (2015)), LOLS chooses which policy to use for each roll-out separately (π^{out} is set inside the foreach loop) and may use a different policy to perform the roll-out for each possible action, but we use the same policy (expert or learned) for all actions to ensure that their costs are calculated consistently. We update this probability p (line 4) at the start of each training iteration, according to the learning rate β. Chang et al. (2015) set this probability to a fixed value throughout all iterations, but we opted to follow SEARN's exponential decay schedule (Daumé III et al., 2009), to increasingly move away from the expert policy as we found it to work better in initial experiments; we discuss this further in section 4.2. By gradually decreasing the use of π^{ref} in the roll-outs, the costs used to train the classifiers are adjusted to their own predictions, thus teaching them to predict actions jointly.

3.1 Expert policy

The expert policy π^{ref} is invoked when assessing possible actions using roll-outs (line 10). Given the sequence of actions already taken $\{a_0 \dots a_t\}$, it acts as an oracle that returns the best possible action

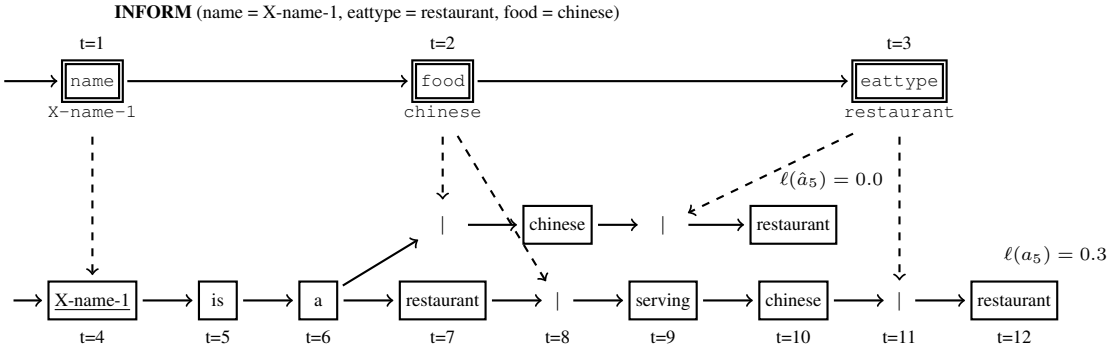

Figure 4: LOLS roll-in (bottom) and roll-out (upper branch) trajectories. Content and word actions are in single and double boxes respectively.

to take, according to the NL reference for the training instance. If a word action is to be predicted by the expert policy we locate the word that makes the sequence (ignoring content actions) best match (i.e. minimize the loss function to) the NL reference. In order to return content actions though, an optimal expert policy would need access to alignments between the MR and the NL reference, which we assume we don't have. Instead, we define a simple heuristic alignment approach that augments the training data with noisy alignments, resulting in a suboptimal expert. LOLS is particularly suitable for this as it can learn with suboptimal expert policies (Chang et al., 2015). This is due to assessing the costs for all possible actions via roll-outs evaluated using the loss function against the (gold) reference, from which the classifier can learn to prefer actions that could contradict those returned by the expert policy. Additionally, in LOLS there is no per-time step mixing of expert and learned policies (i.e. all actions in a single roll-out are predicted by either the expert or the learned policy), which makes designing the expert policy much simpler overall; this is important in tasks like NLG where the optimal action changes drastically when mistakes happen, which is unlike tasks like POS tagging.

The naive alignment used in the expert policy uses the following heuristics. The verbatim values of the MR are simply aligned with their occurrences in the NL reference. The constant values are compared with all unaligned word n-grams in the reference, and aligned with the most similar one according to their character-based Levenshtein distance. For boolean attributes, or attributes with the `dont_care` value, we compare the attribute name with the unaligned n-grams (e.g. aligning `dogs_allowed="yes"` to "allows dogs") instead of the value. Finally, inferred alignments are expanded to unaligned words on the right and left side of the sequence until an already aligned word (or punctuation) is found.

Figure 4 illustrates how examining all possible actions at each time step can overcome the errors in the naive alignments. The content sequence has already been predicted (top of the figure), and at timestep $t = 7$, the classifier returns action $a_7 =$ "restaurant". However, later on the sequence we need to predict words for the attribute `eattype`, for which the classifier (due to the naively aligned training data) is likely to repeat the word "restaurant" leading to an unnatural utterance. So, given this action ($\hat{a}_7 =$ "restaurant"), the resulting sequence has a cost of $\ell(a_7) = 0.3$. By examining all possible actions, the algorithm can surmise that if the classifier stops generating words about the current attribute ($\hat{a}_7 =$ "restaurant"), it would lead to a more natural utterance, and hence a lower loss ($\ell(\hat{a}_7) = 0.0$) which will update the learned policy accordingly. An alternative course of action would be for the classifier to learn to stop generating words about the attribute `eattype` at action $\hat{a}_{12} =$ "restaurant".

3.2 Loss function

In LOLS the loss function is only used to compare complete action sequences against references (line 11). Therefore, it does not need to decompose over individual actions and allows us to use measures like BLEU (Papineni et al., 2002) or ROUGE (Lin, 2004) as loss functions. These measures can consider multiple references for the same MR, which is useful in learning lexical variation, i.e. multiple ways of expressing the same meaning. The loss function is used to assess roll-outs obtained by both learned and expert policies when assessing both action types. In this way, content actions, which are ignored by the loss function, are evaluated based on their impact on word predictions that follow them. When invoking

the expert policy for content actions, the loss function only checks if the reference contains the attributes in the roll-out. We examine the impact of various loss functions in section 4.2.

Additionally, the loss function penalizes actions that are certain to lead to undesirable behavior, i.e. repeating words, producing a termination action without having generated anything, predicting attributes not present in the MR, or attributes whose dictionary does not contain the word(s) that should follow.

3.3 Sequence correction

Another modification we introduced to the original LOLS algorithm is sequence correction in which soon after the first suboptimal action a_t (i.e. not having the minimum cost among those possible) of the roll-in trajectory is encountered (lines 14-18), we attempt to correct the errors encountered so far in the sequence before examining further actions. We allow the algorithm to examine at most E actions ($a_{t+1} \dots a_{t+E}$) following the first suboptimal one (line 16) before re-predicting the roll-in trajectory using the expert policy π^{ref} up to and including the final examined action $a_{1:t+E}$, and using the policy π^{out} to predict the rest of the sequence $a_{t+E:T}$ (lines 17-18). We then proceed with examining the next action a_{t+E+1} of the new corrected sequence. If suboptimal actions are encountered further in the new sequence, sequence correction may again be performed and the sequence re-predicted.

We found this to be helpful in avoiding generating noisy *CSC* training examples. Consider the reference "'X-name-1 is a hotel that allows dogs" and the predicted (incorrect) partial output "X-name-1 is a dog". When assessing the action for the next word, it is likely that predicting "that" or "hotel" will result in lower cost than END_{word} since a loss function such as BLEU or ROUGE would reward the increased word overlap. Assuming that we extract the previous word as a feature in line 12, this would result in learning that predicting "hotel" after "dog" is beneficial. However, this appears to be the case due to the the loss function and the incorrect partial prediction. Correcting the sequence after we encounter the suboptimal word action "dog" to "'X-name-1 is a hotel" will provide the appropriate context for the next action's *CSC*; most of the features Φ_t are extracted from this context. The intuition is similar to the early updates in the incremental perceptron proposed by Collins and Roark (2004).

3.4 Comparison to other Imitation Learning frameworks

In previous subsections, we introduced three modifications to the LOLS framework: 1) the use of SEARN's exponential decay schedule when determining the π^{out} policy, 2) the use of the same policy (expert or learned) for actions in the same step of the sequence, instead of potentially using a different policy for each roll-out, and 3) using sequence correction when encountering suboptimal actions in the sequence.

The main difference of LOLS to other imitation learning frameworks is how the roll-in and roll-out predictions are performed. LOLS performs each roll-in deterministically using the expert policy π^{ref}, while the SEARN and DAGGER (Ross et al., 2011) frameworks roll-in using mixtures of the learned and expert policies; however, partially using the expert policy to roll-in limits those algorithms' capability to learn from the classifiers' mistakes. For roll-out, SEARN stochastically uses a mixture of the learned and expert policies, similarly to LOLS. DAGGER does not employ roll-outs, but uses the expert policy (also referred to as dynamic oracle) to "teach" the correct action to the classifier; this means that it cannot be used to learn using non-decomposable loss functions. Later, a variant of DAGGER, coined V-DAGGER (Vlachos and Clark, 2014), introduced roll-outs to the framework (again using a stochastic mixture of the two policies). AGGREVATE (Ross and Bagnell, 2014) uses a similar roll-in strategy to LOLS, but the expert policy is used exclusively for each roll-out; in the case of NLG, where the expert policy is suboptimal (see section 3.1), this may introduce noise to the classifier.

4 Data and experiments

4.1 Implementation details

We use the adaptive regularization of weight vectors (AROW) algorithm (Crammer et al., 2013) for cost-sensitive classification learning. We train separate content and word classifiers for each predicate, and predicate-attribute combination, respectively. Both types of classifiers (word or content action classifiers) exploit features from the immediately preceding words, attributes and attribute-value pairs, and from

language models estimated on the classifier's corresponding training data. They also consider features based on the input MR, such as which attributes and values have already been mentioned and which have not, and, specifically for the word classifiers, whether a word partly or wholly expresses a value.

During training, we also prevent the classifiers from generating more actions than what has been observed in the training data. Finally, the NL utterances are generated in a "delexicalized" form, but we replace the variables with the corresponding values in post-processing; this is the reverse procedure of replacing verbatim values with variables as shown in Figures 1 and 2. The same system is applied to all datasets discussed in the following sections, and its code will be publicly available upon publication.

4.2 Experiments with the SF datasets

The two SF datasets were created by Wen et al. (2015) and contain MRs about hotels and restaurants respectively.[2] The hotel dataset consists of 5,192 MRs with a single NL reference each. They contain 8 predicates (`inform`, `confirm`, etc.) and 12 attributes, some of them unique to each dataset. Each attribute can take at most one value (or be empty). Following Wen et al. (2015) we partitioned the data into a training, validation, and testing set in a 3:1:1 ratio. We also created multiple references for each validation and test MR as they did, by grouping all delexicalized references that correspond to the same delexicalized MR. We then repopulate the references according to the MR's values.

Figure 5 shows heat maps of BLEU-4 (top set) and ROUGE-4 (bottom set) scores our system achieved on the validation set of the SF hotel dataset for different values of the learning rate β and the sequence correction parameter E; the notation *"std"* denotes that no sequence correction was used in that configuration. The heat maps are also grouped by which loss function was used in each configuration. By comparing the best configurations, we can conclude that using sequence correction leads to slightly better results (a 1.56 difference on average in BLEU). In figure 6 we show how the learned policy is improved with each epoch of LOLS ($\pi_0 \ldots \pi_3$); note that figure 5 shows the best epoch of each configuration.

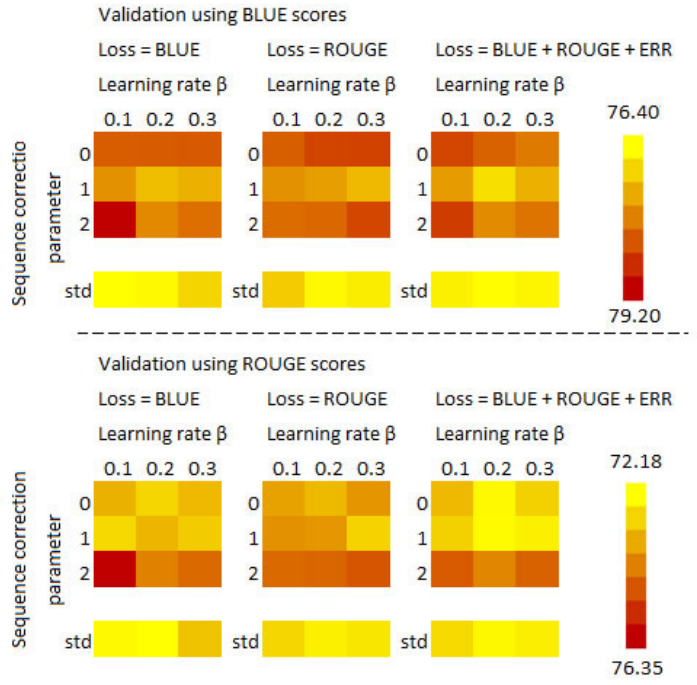

Figure 5: Result heat map for SF hotels across different parameters.

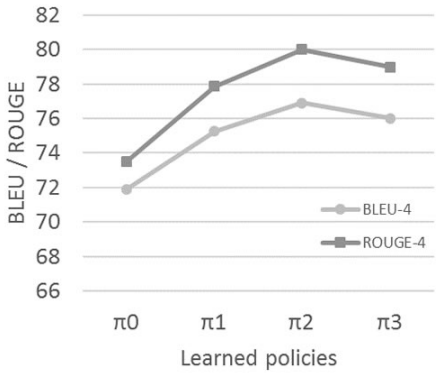

Figure 6: Result heat map for SF hotels across different parameters.

As loss functions to LOLS we tried BLEU-4, ROUGE-4, and the harmonic mean of BLEU-4, ROUGE-4 and ERR(%). In essence, BLEU measures the precision of the sentence's content while ROUGE measures the recall. We opted to use ROUGE-4 rather than other variants of ROUGE since it has been shown to

[2] SF is freely available for download at: https://www.repository.cam.ac.uk/handle/1810/251304

correlate better with human assessments in summarization tasks (Graham, 2015). ERR(%) measures what portion of the MRs is not expressed in the NL sentence; it is defined as the number of attribute-value pairs in the MR that are not expressed in the generated NL sentence to the total number of attribute-value pairs in the MR. We can map the values of the MR to the NL sentence similarly to how we calculate the naive alignments (see section 3.1). By comparing these loss functions on the heat maps of Figure 5, we find that there is very little difference between them (0.12 difference on BLEU, and 0.36 on ROUGE). We decided to use the harmonic mean in our experiments since it captures more information.

Additionally, we tried setting a fixed value to probability p (see line 4 in algorithm 2) as per Chang et al. (2015) instead of using SEARN's exponential decay schedule (Daumé III et al., 2009). When comparing the best configurations on the validation set (with no sequence correction), using a fixed value seems to be slightly worse than using an exponential decay schedule (0.89 BLEU difference in SF hotel).

To assess the impact of naive alignments (see section 3.1) on our system we tried initializing the policy π_0 on random alignments. In this case, the values of the MR are aligned with words in the NL reference randomly. Subsequently, the random alignments are expanded alternatively to unaligned words similarly to naive alignment expansion as described in section 3.1. The best configuration with random alignments, as expected, performs worse to using naive alignments (25.84 difference in BLEU), but LOLS still improves on the initial policy (5.49 improvement on π_0). While all results reported above are on the SF hotel dataset, similar conclusions can be drawn for the other datasets we examined.

	SF RESTAURANT								
	FULL TEST (1040 MRS)			UNIQUE (158 MRS)			NON-OVERLAPPING (13 MRS)		
	BLEU	ROUGE	ERR(%)	BLEU	ROUGE	ERR(%)	BLEU	ROUGE	ERR(%)
WEN	74.50	77.75	2.54	52.97	43.52	6.29	27.04	20.44	10.38
LOLS	66.01	64.56	0.15	49.44	38.52	0.58	28.21	21.47	0.00

	SF HOTEL								
	FULL TEST (1076 MRS)			UNIQUE (96 MRS)			NON-OVERLAPPING (11 MRS)		
	BLEU	ROUGE	ERR(%)	BLEU	ROUGE	ERR(%)	BLEU	ROUGE	ERR(%)
WEN	86.54	84.36	0.88	66.37	56.19	3.99	37.24	27.27	6.82
LOLS	80.00	76.88	0.25	68.65	68.37	0.52	33.31	27.01	3.63

Table 1: Results on the SF datasets.

Finally, we use the test set to compare against the results of Wen et al. (2015), while setting the parameters based on the validation set results. Our test set results are summarized in Table 1; we report BLEU-4, ROUGE-4, and ERR(%). The BLEU-4 and ERR(%) scores of Wen et al. (2015) have been updated after personal communication with them; both are now computed using the same implementation of the measures. The scores are calculated on the full test set, as well as two smaller subsets. The first subset consists exclusively of the unique MRs in the test set (i.e. no MR appears more than once in the unique subset). The results in the unique subset may be more indicative of a system's performance as the distribution of MRs in the test set is unbalanced, e.g. an extreme case in the full test set, is the MR `goodbye()` which appears 176 times. Comparing the two systems, we see that LOLS performs worse in terms of BLEU and ROUGE, but also has lower ERR(%); this indicates that our system tries to include more attributes in the NL sentences. However, when looking at the scores in the unique MR subset, the difference between the two systems narrows, with almost no difference in the SF hotel dataset.

Examining the test set further, we noticed that there is a big overlap between the training, validation and test sets when we consider the MRs in their delexicalized form (see sction 2). Given that both our and Wen et al. (2015)'s models are trained on the delexicalized forms of the training MRs that means that the evaluation is, in part, performed on MRs that have already been encountered during training. In an effort to more accurately assess how the models generalize to unseen input, we define a second subset of the test set, consisting of the unique MRs whose delexicalized version does not appear in either the training nor validation data (i.e. non-overlapping MRs). As it turns out, these are only 11 and 13 non-overlapping MRs in the SF hotel and restaurant datasets respectively. Comparing the two systems in this second subset shows little difference between them; however we cannot draw any conclusions as the subset is very small. We show below some output utterances from Wen et al. (2015)'s system and ours:

MR: inform(type=restaurant,count='239',good_for_meal=dont_care,food=dont_care)

WEN: I have found 239 restaurants serve any meal if the food type is not an issue.

LOLS: There are 239 restaurants if you do not care about the meal and food.

MR: inform(name='kean hotel',has_internet='no')

WEN: We're sorry, the Kean hotel does not have access to the internet.

LOLS: We're sorry, the Kean hotel does not have internet.

4.3 Experiments with the BAGEL dataset

The BAGEL dataset was developed by Mairesse et al. (2010) and contains 202 MRs, each corresponding to 2 NL references.[3] The dataset describes restaurants, and contains one predicate (`inform`) and 8 attributes (see Figure 1). Each attribute can take multiple values, which are either verbatim strings or constants.

	BLEU	ROUGE	NIST	ERR(%)
DUSEK				
Basic perceptron	52.35	46.43	4.777	23.90
+ Diff-tree upd.	56.75	50.82	5.137	25.10
+ Future promise	56.71	48.08	5.441	24.16
LOLS	54.22	47.39	5.547	16.66

Table 2: Results on the BAGEL dataset.

In this section we are comparing against Dušek and Jurcícek (2015), who based their work on the BAGEL dataset. We include results from three variations of their system, the first using only a perceptron to score the output, the second augmented with differing tree updates and the last with future promise estimations. Following their work, we use 10-fold cross validation, with no overlap between MRs in the training set and the test set; we obtained and use the same folds. Their scores have been recalculated using the same implementation of the measures we use. The BAGEL dataset has no separate development set, so we tuned our system's parameters using 10-fold-cross-validation as well.

Table 2 shows our results; we report BLEU-4, ROUGE-4, NIST (Doddington, 2002), and ERR(%). We have achieved better NIST scores than Dušek and Jurcícek (2015)'s, with slightly worse BLEU-4 and ROUGE-4 scores. Our lower ERR(%) suggests that our system includes more information (i.e. attributes-value pairs) into the sentences, which may account for the slightly worse BLEU and ROUGE scores. Another explanation for Dušek and Jurcícek (2015)'s performance is, as stated before, that their surface realization is performed using rules based on the frames of an English-Czech dependency treebank.

4.4 Human evaluation

To better assess the perceived quality of our system's output, we showed pairs of MRs and generated utterances for all datasets to 202 human judges, not involved in the work of this article. They were all fluent, though not native, English speakers. We generated 1076 + 1040 + 202 texts from the MRs of the SF hotel, SF restaurant, and BAGEL data sets using our NLG system, and also obtained the respective texts that Wen et al. (2015) and Dušek and Jurcícek (2015) generated in their work using their best configurations. Each MR and generated utterance was shown to (at least) three human judges, in random order. For each pair, the judges were asked to either score the generated utterance in terms of fluency, and informativeness; a scale from 1 to 6 was used. Each judge scored fluency and informativeness on separate sets of texts, to minimize correlation between the criteria. Among the texts that every judge scored, was a small subset of texts whose quality in regards to each other was known beforehand. We used that subset to filter out unreliable human judges that did not score these texts as expected.

	BAGEL		SF RESTAURANT		SF HOTEL	
	Dušek and Jurcícek (2015) + Future promise	LOLS	Wen et al. (2015)	LOLS	Wen et al. (2015)	LOLS
Fluency	5.15*	4.79*	4.49	4.23	4.41	4.68
Informativeness	4.53*	5.24*	5.29	5.36	5.36	5.19

Table 3: Human evaluation on BAGEL and SF datasets.

[3] BAGEL is freely available for download at: http://farm2.user.srcf.net/research/bagel/

Table 3 shows the results of the human evaluation, averaged over all the texts. In the BAGEL dataset, our system performs slightly worse in terms of fluency, but is perceived as more informative. In the SF restaurant dataset, our system performs worse in terms of fluency and better in informativeness, while in the SF hotel dataset, our system performs better in fluency and worse in informativeness. However, no statistically significant difference was detected in the SF datasets for both of these criteria.[4] Table 4, shows the results if we isolate the human judgements for the subset of non-overlapping MRs (see section 4.2); we see that the differences between the systems are much clearer but again we must state that no safe conclusions can be drawn from that small a subset.

	SF RESTAURANT		SF HOTEL	
	Wen et al. (2015)	LOLS	Wen et al. (2015)	LOLS
Fluency	3.53	3.23	2.38	4.22
Informativeness	4.90	5.23	4.65	4.30

Table 4: Human evaluation on the non-overlapping MRs of the SF datasets.

5 Related work

Apart from the works we compared against, only Konstas and Lapata (2013) and Mei et al. (2016) do not need pre-aligned data. Konstas and Lapata (2013)'s approach incorporates the work of Liang et al. (2009) that learns a generative semi-Markov model to calculate the alignments. We note that this alignment model is developed on the datasets considered, and does not generalize equally well to other datasets (Angeli et al., 2010). On the other hand, the naive alignments we infer are much simpler and we improve them by joint learning of word and content action prediction with respect to the sentence-level evaluation via BLEU and ROUGE. Concurrently, Mei et al. (2016) introduced an encoder-aligner-decoder model to perform content selection and surface realization without pre-aligned data. Their work employs bidirectional LSTM-RNN models, similarly to the work of Wen et al. (2015), and a coarse-to-fine aligner. Unfortunately, they do not report results in the datasets we performed our evaluation on, do not compare against Wen et al. (2015), and their code was unavailable when we were preparing this article.

Imitation learning algorithms for structured prediction have been applied successfully to a variety of tasks, such as dependency parsing (Goldberg and Nivre, 2013) and dynamic feature selection (He et al., 2013). Vlachos and Clark (2014) applied a variant of DAGGER (Ross et al., 2011) to learning a semantic parser from unaligned training examples, which is the reverse task to NLG, i.e. predicting the MR given the NL utterance. To circumvent the lack of alignment information they resorted to defining a randomized expert policy similar to the heuristic one we define, but NLG poses a greater challenge since the output space is all English sentences possible given the vocabulary considered.

Finally, we believe that the main benefit of our imitation learning approach, namely that it is able to learn using a non-decomposable loss function, is orthogonal to using continuous representations such as the hidden state and memory cell in the LSTM of Wen et al. (2015). Recent work by Ranzato et al. (2016) showed how RNNs can be trained at the sequence level (as opposed to the word level) with non-decomposable loss functions in the context of machine translation using imitation learning, and such an approach would also be applicable to NLG.

6 Conclusions

In this work, we adapted the Locally Optimal Learning to Search algorithm (Chang et al., 2015) to learn an NLG system from unaligned training data, focusing on its ability to learn with non-decomposable loss functions and suboptimal expert policies. We compared our approach to two recently proposed approaches for learning NLG from unaligned data on datasets they were developed on (three datasets across two domains) and achieved comparable results in automatic evaluation. We also performed human evaluation which showed that our system performs comparably, though its fluency requires improvement on MRs with multiple attributes.

[4] We performed Analysis of Variance (ANOVA) and post-hoc Tukey tests ($\alpha = 0.05$); * denotes statistical significance.

Acknowledgements

This research is supported by the EPSRC grant Diligent (EP/M005429/1). We would like to thank Tsung-Hsien Wen and Ondřej Dušek for their help on the evaluation, Lucia Specia for her advice on the evaluation, and anyone who participated in the human evaluation. Finally, we would like to thank our anonymous reviewers for their helpful comments that helped improve the article.

References

I. Androutsopoulos, G. Lampouras, and D. Galanis. 2013. Generating natural language descriptions from OWL ontologies: the NaturalOWL system. *Journal of Artificial Intelligence Research*, 48(1):671–715.

G. Angeli, P. Liang, and D. Klein. 2010. A simple domain-independent probabilistic approach to generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2010)*, pages 502–512, Cambridge, MA.

K. Bontcheva and Y. Wilks. 2004. Automatic report generation from ontologies: the MIAKT approach. In *Proceedings of the 9th International Conference on Applications of Natural Language to Information Systems (NLDB 2004)*, pages 324–335, Manchester, UK.

Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daumé III, and John Langford. 2015. Learning to search better than your teacher. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, pages 111–118. Association for Computational Linguistics.

Koby Crammer, Alex Kulesza, and Mark Dredze. 2013. Adaptive regularization of weight vectors. *Machine Learning*, 91:155–187.

Robert Dale, Sabine Geldof, and Jean-Philippe Prost. 2003. Coral : Using natural language generation for navigational assistance. In Michael J. Oudshoorn, editor, *Twenty-Sixth Australasian Computer Science Conference (ACSC2003)*, volume 16 of *CRPIT*, pages 35–44, Adelaide, Australia. ACS.

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

Nina Dethlefs, Helen Hastie, Heriberto Cuayáhuitl, and Oliver Lemon. 2013. Conditional random fields for responsive surface realisation using global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1254–1263, Sofia, Bulgaria, August. Association for Computational Linguistics.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the Second International Conference on Human Language Technology Research*, HLT 2002, pages 138–145, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Ondřej Dušek and Filip Jurcícek. 2015. Training a natural language generator from unaligned data. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 451–461.

Yoav Goldberg and Joakim Nivre. 2013. Training deterministic parsers with non-deterministic oracles. *Transactions of the association for Computational Linguistics*, 1:403–414.

Yvette Graham. 2015. Re-evaluating automatic summarization with bleu and 192 shades of rouge. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 128–137, Lisbon, Portugal, September. Association for Computational Linguistics.

He He, Hal Daumé III, and Jason Eisner. 2013. Dynamic feature selection for dependency parsing. In *Empirical Methods in Natural Language Processing*.

Ioannis Konstas and Mirella Lapata. 2013. A global model for concept-to-text generation. *Journal of Artificial Intelligence Research*, 48:305–346.

P. Liang, M. Jordan, and D. Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of the 47th Meeting of the Association of Computational Linguistics and 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL/AFNLP '09)*, pages 91–99, Suntec, Singapore.

C.W. Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Proceedings of ACL-04 Workshop: Text Summarization Branches Out*.

François Mairesse, Milica Gašić, Filip Jurčíček, Simon Keizer, Blaise Thomson, Kai Yu, and Steve Young. 2010. Phrase-based statistical language generation using graphical models and active learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL 2010, pages 1552–1561, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. *Proceedings of NAACL*.

K. Papineni, S. Roukos, T. Ward, and W. J. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, pages 311–318, Philadelphia, PA.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *Proceedings of the 2016 International Conference on Learning Representations*.

Ehud Reiter, Somayajulu Sripada, Jim Hunter, and Ian Davy. 2005. Choosing words in computer-generated weather forecasts. *Artificial Intelligence*, 167:137–169.

Stéphane Ross and J. Andrew Bagnell. 2014. Reinforcement and imitation learning via interactive no-regret learning. *CoRR*, abs/1406.5979.

Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. 2011. 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.

Andreas Vlachos and Stephen Clark. 2014. A new corpus and imitation learning framework for context-dependent semantic parsing. *Transactions of the Association for Computational Linguistics*, 2:547–559, December.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, September.