SHEF-MIME: Word-level Quality Estimation Using Imitation Learning

Daniel Beck Andreas Vlachos Gustavo H. Paetzold Lucia Specia
Department of Computer Science
University of Sheffield, UK
{debeck1,a.vlachos,gpaetzold1,l.specia}@sheffield.ac.uk

Abstract

We describe University of Sheffield's submission to the word-level Quality Estimation shared task. Our system is based on imitation learning, an approach to structured prediction which relies on a classifier trained on data generated appropriately to ameliorate error propagation. Compared to other structure prediction approaches such as conditional random fields, it allows the use of arbitrary information from previous tag predictions and the use of non-decomposable loss functions over the structure. We explore these two aspects in our submission while using the baseline features provided by the shared task organisers. Our system outperformed the conditional random field baseline while using the same feature set.

1 Introduction

Quality estimation (QE) models aim at predicting the quality of machine translated (MT) text (Blatz et al., 2004; Specia et al., 2009). This prediction can be at several levels, including word-, sentence- and document-level. In this paper we focus on our submission to the word-level QE WMT 2016 shared task, where the goal is to assign quality labels to each word of the output of an MT system.

Word-level QE is traditionally treated as a structured prediction problem, similar to part-of-speech (POS) tagging. The baseline model used in the shared task employs a Conditional Random Field (CRF) (Lafferty et al., 2001) with a set of baseline features. Our system uses a linear classification model trained with imitation learning (Daumé III et al., 2009; Ross et al., 2011). Compared to the baseline approach that uses a CRF, imitation learning has two benefits:

- We can directly use the proposed evaluation metric as the loss to be minimised during training;
- It allows using richer information from previous label predictions in the sentence.

Our primary goal with our submissions was to examine if the above benefits would result in better accuracy than that for the CRF. For this reason, we did not perform any feature engineering: we made use instead of the same features as the baseline model. Both our submissions outperformed the baseline, showing that there is still room for improvements in terms of modelling, beyond feature engineering.

2 Imitation Learning

A naive, but simple way to perform word-level QE (and any word tagging problem) is to use an off-the-shelf classifier to tag each word extracting features based on the sentence. These usually include features derived from the word being tagged and its context. The main difference between this approach and structure prediction methods is that it treats each tag prediction as independent from each other, ignoring the structure behind the full tag sequence for the sentence.

If we treat the observed sentence as a *sequence* of words (from left to right) then we can modify the above approach to perform a sequence of *actions*, which in this case are tag predictions. This setting allows us to incorporate structural information in the classifier by using features based on previous tag predictions. For instance, let us assume that we are trying to predict the tag t_i for word w_i. A simple classifier can use features derived from w_i and also any other words in the sentence. By framing this as a sequence, it can also use features extracted from the previously predicted tags $t_{\{1:i-1\}}$.

This approach to incorporating structural information suffers from an important problem: during training it assumes the features based on previous tags come from a perfectly predicted sequence (the gold standard). However, during testing this sequence will be built by the classifier, thus likely to contain errors. This mismatch between training and test time features is likely to hurt the overall performance since the classifier is not trained to *recover* from its errors, resulting in error propagation.

Imitation learning (also referred to as search-based structure prediction) is a general class of methods that attempt to solve this problem. The main idea is to first train the classifier using the gold standard tags, and then generate examples by using the trained classifier to re-predict the training set and update the classifier using these new examples. The example generation and classification training is usually repeated. The key point in this procedure is that because the examples are generated in the training set we are able to query the gold standard for the correct tags. So, if the classifier makes a wrong prediction at word w_i we can teach it to recover from this error at word w_{i+1} by simply checking the gold standard for the right tag.

In the imitation learning literature the sequence of predictions is referred to as *trajectory*, which is obtained by running a *policy* on the input. Three kinds of policy are commonly considered:

- *expert policy*, which returns the correct prediction according to the gold standard and thus can only be used during training,
- *learned policy*, which queries the trained classifier for its prediction,
- and *stochastic mixture* between *expert* and *learned*.

The most commonly used imitation learning algorithm, DAGGER (Ross et al., 2011), initially uses the expert policy to train a classifier and subsequently uses a stochastic mixture policy to generate examples based on a 0/1 loss on the current tag prediction with respect to the expert policy (which returns the correct tag according to the gold standard). This idea can be extended by, instead of taking the 0/1 loss, applying the same stochastic policy until the end of the sentence and calculating a loss over the entire tag sequence with respect to the gold standard. This generates a *cost-sensitive* classification training example and allows the algorithm to use arbitrary, potentially non-decomposable losses during training. This is the approach used by Vlachos and Clark (2014) and which is employed in our submission (henceforth called V-DAGGER). Its main advantage is that it allows us to use a loss based on the final shared task evaluation metric. The latter is the F-measure on 'OK' labels times F-measure on 'BAD' labels, which we turn into a loss by subtracting it from 1.

Algorithm 1 V-DAGGER algorithm

```
Input  training instances $\mathcal{S}$, expert policy $\pi^*$, loss
       function $\ell$, learning rate $\beta$, cost-sensitive clas-
       sifier $CSC$, learning iterations $N$
Output  learned policy $\pi_N$
 1: CSC instances $E = \emptyset$
 2: for $i = 1$ to $N$ do
 3:     $p = (1-\beta)^{i-1}$
 4:     current policy $\pi = p\pi^* + (1-p)\pi_i$
 5:     for $s \in \mathcal{S}$ do
 6:                          ▷ assuming $T$ is the length of $s$
 7:         predict $\pi(s) = \hat{y}_{1:T}$
 8:         for $\hat{y}_t \in \pi(s)$ do
 9:             get observ. features $\phi_t^o = f(s)$
10:             get struct. features $\phi_t^s = f(\hat{y}_{1:t-1})$
11:             concat features $\phi_t = \phi_t^o || \phi_t^s$
12:             for all possible actions $y_t^j$ do
13:                                  ▷ predict subsequent actions
14:                 $y'_{t+1:T} = \pi(s; \hat{y}_{1:t-1}, y_t^j)$
15:                                               ▷ assess cost
16:                 $c_t^j = \ell(\hat{y}_{1:t-1}, y_t^j, y'_{t+1:T})$
17:             end for
18:             $E = E \cup (\phi_t, c_t)$
19:         end for
20:     end for
21:     learn $\pi_i = CSC(E)$
22: end for
```

Algorithm 1, which is replicated from (Vlachos and Clark, 2014), details V-DAGGER. At line 4 the algorithm selects a policy to predict the tags (line 7). In the first iteration it is just the expert policy, but from the second iteration onwards it becomes a stochastic mixture of the expert and learned policies. The cost-sensitive instances are generated by iterating over each word in the instance (line 8), extracting features from the instance itself (line 9) and the previously predicted tags (line 10) and estimating a cost for each possible tag (lines 12-17). These instances are then used to train a cost-sensitive classifier, which becomes the new learned policy (line 21). The whole procedure is repeated until a desired iteration budget N is reached.

The feature extraction step at lines 9 and 10 can be made in a single step. We chose to split it between *observed* and *structural* features to emphasise the difference between our method and the CRF baseline. While CRFs in theory can employ any kind of structural features, they are usually restricted to consider only the previous tag for efficiency (1st order Markov assumption).

3 Experimental Settings

The shared task dataset consists of 15k sentences translated from English to German using an MT system and post-edited by professional translators. The post-edited version of each sentence is used to obtain quality tags for each word in the MT output. In this shared task version, two tags are employed: an 'OK' tag means the word is correct and a 'BAD' tag corresponds to a word that needs a post-editing action (either deletion, substitution or the insertion of a new word). The official split corresponds to 12k, 1k and 2k for training, development and test sets.

Model Following (Vlachos and Clark, 2014), we use AROW (Crammer et al., 2009) for cost-sensitive classification learning. The loss function is based on the official shared task evaluation metric: $\ell = 1 - [F(\text{OK}) \times F(\text{BAD})]$, where F is the tag F-measure at the sentence level.

We experimented with two values for the learning rate β and we submitted the best model found for each value. The first value is 0.3, which is the same used by Vlachos and Clark (2014). The second one is 1.0, which essentially means we use the expert policy only in the first iteration, switching to using the learned policy afterwards.

For each setting we run up to 10 iterations of imitation learning on the training set and evaluate the score on the dev set after each iteration. We select our model in each learning rate setting by choosing the one which performs the best on the dev set. For $\beta = 1.0$ this was achieved after 10 iterations, but for $\beta = 0.3$ the best model was the one obtained after the 6th iteration.

Observed features The features based on the observed instance are the same 22 used in the baseline provided by the task organisers. Given a word w_i in the MT output, these features are defined below:

- Word and context features:
  - w_i (the word itself)
  - w_{i-1}
  - w_{i+1}
  - w_i^{src} (the aligned word in the source)
  - w_{i-1}^{src}
  - w_{i+1}^{src}
- Sentence features:
  - Number of tokens in the source sentence
  - Number of tokens in the target sentence
  - Source/target token count ratio
- Binary indicators:
  - w_i is a stopword
  - w_i is a punctuation mark
  - w_i is a proper noun
  - w_i is a digit
- Language model features:
  - Size of largest n-gram with frequency $>$ 0 starting with w_i
  - Size of largest n-gram with frequency $>$ 0 ending with w_i
  - Size of largest n-gram with frequency $>$ 0 starting with w_i^{src}
  - Size of largest n-gram with frequency $>$ 0 ending with w_i^{src}
  - Backoff behavior starting from w_i
  - Backoff behavior starting from w_{i-1}
  - Backoff behavior starting from w_{i+1}
- POS tag features:
  - The POS tag of w_i
  - The POS tag of w_i^{src}

The language model backoff behavior features were calculated following the approach in (Raybaud et al., 2011).

Structural features As explained in Section 2, a key advantage of imitation learning is the ability to use arbitrary information from previous predictions. Our submission explores this by defining a set of features based on this information. Taking t_i as the tag to be predicted for the current word, these features are defined in the following way:

- Previous tags:
  - t_{i-1}
  - t_{i-2}
  - t_{i-3}
- Previous tag n-grams:
  - $t_{i-2}||t_{i-1}$ (tag bigram)
  - $t_{i-3}||t_{i-2}||t_{i-1}$ (tag trigram)
- Total number of 'BAD' tags in $t_{1:t-1}$

Results Table 1 shows the official shared task results for the baseline and our systems, in terms of F1-MULT, the official evaluation metric, and also F1 for each of the classes. We report two versions for our submissions: the official one, which had an implementation bug[1] and a new version after the bug fix.

Both official submissions outperformed the baseline, which is an encouraging result considering that we used the same set of features as the baseline. The submission which employed $\beta = 1$ performed the best between the two. This is in line with the observations of Ross et al. (2011) in similar sequential tagging tasks. This setting allows the classifier to move away from using the expert policy as soon as the first classifier is trained.

	F1-BAD	F1-OK	F1-MULT
Baseline (CRF)	0.3682	0.8800	0.3240
Official submission			
$N = 6, \beta = 0.3$	0.3909	0.8450	0.3303
$N = 10, \beta = 1.0$	0.4029	0.8392	0.3380
Fixed version			
$N = 9, \beta = 0.3$	0.3996	0.8435	0.3370
$N = 9, \beta = 1.0$	0.4072	0.8415	0.3426

Table 1: Official shared task results.

Analysis To obtain further insights about the benefits of imitation learning for this task we performed additional experiments with different settings. In Table 2 we compare our systems with a system trained using a single round of training (called *exact imitation*), which corresponds to using the same classifier trained only on the gold standard tags. We can see that imitation learning improves over this setting substantially.

Table 2 also shows results obtained using the original DAGGER algorithm, which uses a single 0/1-loss per tag. While DAGGER improves results over the exact imitation setting, it is outperformed by V-DAGGER. This is due to the ability of V-DAGGER to incorporate the task loss into its training procedure[2].

In Figure 1 we compare how the F1-MULT scores evolve through the imitation learning iterations for both DAGGER and V-DAGGER. Even though the performance of V-DAGGER fluctuates more than that of DAGGER, it is consistently better for both development and test sets.

Finally, we also compare our systems with simpler versions using a smaller set of structural features. The findings, presented in Table 3, show an interesting trend. The systems do not seem to benefit from the additional structural information available in imitation learning and even a system with no information at all ("None" in Table 3) outperforms the baseline. We speculate that this is because the task only deals with a linear chain of binary labels, which makes the structure much less informative compared to the observed features.

	F1-BAD	F1-OK	F1-MULT
Exact imitation	0.2503	0.8855	0.2217
DAGGER			
$N = 10, \beta = 0.3$	0.3322	0.8483	0.2818
$N = 4, \beta = 1.0$	0.3307	0.8758	0.2897
V-DAGGER			
$N = 9, \beta = 0.3$	0.3996	0.8435	0.3370
$N = 9, \beta = 1.0$	0.4072	0.8415	0.3426

Table 2: Comparison between our systems (V-DAGGER), exact imitation and DAGGER on the test data.

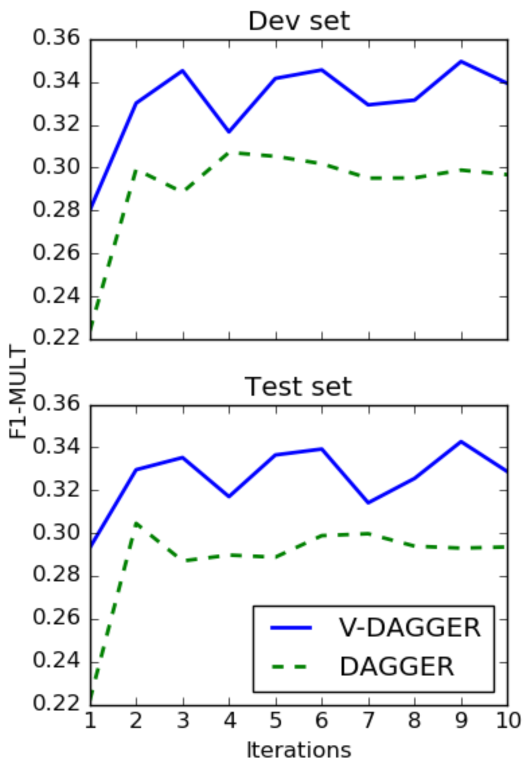

Figure 1: Metric curves for DAGGER and V-DAGGER over the official development and test sets. Both settings use $\beta = 1.0$.

[1]The structural feature t_{i-1} was not computed properly.

[2]Formally, our loss is not exactly the same as the official shared task evaluation metric since the former is measured at the sentence level and the latter at the corpus level. Nevertheless, the loss in V-DAGGER is much closer to the official metric than the 0/1-loss used by DAGGER.

	F1-BAD	F1-OK	F1-MULT
$\beta = 0.3$			
None	0.3948	0.8536	0.3370
t_{i-1}	0.3873	0.8393	0.3251
$t_{i-1} + t_{i-2} \| t_{i-1}$	0.3991	0.8439	0.3368
All	0.3996	0.8435	0.3370
$\beta = 1.0$			
None	0.3979	0.8530	0.3394
t_{i-1}	0.4089	0.8436	0.3449
$t_{i-1} + t_{i-2} \| t_{i-1}$	0.4094	0.8429	0.3451
All	0.4072	0.8415	0.3426

Table 3: Comparison between V-DAGGER systems using different structural feature sets. All models use the full set of observed features.

4 Conclusions

We presented the first attempt to use imitation learning for the word-level QE task. One of the main strengths of our model is its ability to employ non-decomposable loss functions during the training procedure. As our analysis shows, this was a key reason behind the positive results of our submissions with respect to the baseline system, since it allowed us to define a loss function using the official shared task evaluation metric. The proposed method also allows the use of arbitrary information from the predicted structure, although its impact was much less noticeable for this task.

The framework presented in this paper could be enhanced by going beyond the QE task and applying actions in subsequent tasks, such as automatic post-editing. Since this framework allows for arbitrary loss functions it could be trained by optimising MT metrics like BLEU or TER. The challenge in this case is how to derive expert policies: unlike simple word tagging, multiple action sequences could result in the same post-edited sentence.

Acknowledgements

This work was supported by CNPq (project SwB 237999/2012-9, Daniel Beck), the QT21 project (H2020 No. 645452, Lucia Specia) and the EPSRC grant Diligent (EP/M005429/1, Andreas Vlachos).

References

John Blatz, Erin Fitzgerald, and George Foster. 2004. Confidence estimation for machine translation. In *Proceedings of the 20th Conference on Computational Linguistics*, pages 315–321.

Koby Crammer, Alex Kulesza, and Mark Dredze. 2009. Adaptive Regularization of Weight Vectors. In *Advances in Neural Information Processing Systems*, pages 1–9.

Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Sylvain Raybaud, David Langlois, and Kamel Smali. 2011. This sentence is wrong. Detecting errors in machine-translated sentences. *Machine Translation*, (1).

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. 2011. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. In *Proceedings of AISTATS*, volume 15, pages 627–635.

Lucia Specia, Nicola Cancedda, Marc Dymetman, Marco Turchi, and Nello Cristianini. 2009. Estimating the sentence-level quality of machine translation systems. In *Proceedings of EAMT*, pages 28–35.

Andreas Vlachos and Stephen Clark. 2014. A New Corpus for Context-Dependent Semantic Parsing. *Transactions of the Association for Computational Linguistics*, 2:547–559.