Guiding Neural Machine Translation Decoding with External Knowledge

Rajen Chatterjee$^{(1,2)}$, **Matteo Negri**$^{(2)}$, **Marco Turchi**$^{(2)}$,
Marcello Federico$^{(2)}$, **Lucia Specia**$^{(3)}$, and **Frédéric Blain**$^{(3)}$
$^{(1)}$University of Trento, Trento, Italy
$^{(2)}$Fondazione Bruno Kessler, Trento, Italy
$^{(3)}$University of Sheffield, Sheffield, UK
{chatterjee,negri,turchi,federico}@fbk.eu
{l.specia,f.blain}@sheffield.ac.uk

Abstract

Differently from the phrase-based paradigm, neural machine translation (NMT) operates on word and sentence representations in a continuous space. This makes the decoding process not only more difficult to interpret, but also harder to influence with external knowledge. For the latter problem, effective solutions like the XML-markup used by phrase-based models to inject fixed translation options as constraints at decoding time are not yet available. We propose a "guide" mechanism that enhances an existing NMT decoder with the ability to prioritize and adequately handle translation options presented in the form of XML annotations of source words. Positive results obtained in two different translation tasks indicate the effectiveness of our approach.

1 Introduction

The need to enforce fixed translations of certain source words is a well known problem in machine translation (MT). For instance, this is an issue in application scenarios in which the translation process has to comply with specific terminology and/or style guides. In such situations it is generally necessary to consider external resources to guide the decoder in order to ensure consistency or meet other specific requirements. Terminology lists, which provide the decoder with the expected translations of specific words or phrases, are a typical example of external knowledge used to guide the process to meet such constraints. Meeting predefined constraints, however, does not represent the only case in which an external guidance can support decoding. In ensemble MT architectures, for example, the output of a translation system specialised in handling specific phenomena (*e.g.* numbers or dates) can be used to guide another decoder without changing its underlying model.

Phrase-based statistical MT (PBSMT), which explicitly manipulates symbolic representations of the basic constituents (phrases) in the source and target languages, provides solutions to address these needs. For instance, the XML markup implemented in the Moses toolkit (Koehn et al., 2007) allows one to supply the expected translations to the decoder in the form of tags surrounding the corresponding source phrases.

To our knowledge, solutions to this problem are not yet available for neural machine translation (NMT), which has recently emerged as the dominant approach for MT. In particular, no work has been done to address the needs of the translation industry, in which language service providers usually receive translation requests that must be satisfied in short time, often taking into account external knowledge that defines specific customers' constraints. In this case, the time-consuming retraining routines of NMT are not viable, thus making methods to inject external knowledge without retraining of paramount importance.

To address this gap, we investigate problems arising from the fact that NMT operates on implicit word and sentence representations in a continuous space, which makes influencing the process with external knowledge more complex. In particular, we attempt to answer the following questions: *i)* How to enforce the presence of a given translation recommendation in the decoder's output? *ii)* How to place these word(s) in the right position? *iii)* How to guide the translation of out-of-vocabulary terms?

Our solution extends an existing NMT decoder (Sennrich et al., 2016a) by introducing the possibility to guide the translation process with constraints provided as XML annotations of the source words with the corresponding translation options. The guidance mechanism supervises the process, generating the final output with the expected translations, in the right place, including cases of external words unknown to the model.

To test our approach, we experiment in two scenarios that pose different challenges to NMT. The first one is a translation task in which source sentences contain XML-annotated domain-specific terms. The presence of few annotated terms poses fewer constraints to the decoder in generating the output sentence. The second scenario is an automatic post-editing (APE) task, in which the NMT model is trained to translate "monolingually" from draft machine-translated sentences into human-quality post-edits. The external guidance is provided by word-level quality judgements (Blatz et al., 2004) indicating the "good" words in the machine-translated sentence that should be kept in the final APE output. In this case, the large number of "good" words already present in the original MT output poses more constraints to the decoding process. In both scenarios, our guidance mechanism achieves significant performance gains over the original NMT decoder.

2 Related Work

In PBSMT, the injection of external knowledge in the decoder is usually handled with the so-called XML markup, a technique used to guide the decoder by supplying the desired translation for some of the source phrases. The supplied translation choice can be injected in the output by using different strategies, all rather straightforward. Examples include manipulating the phrase table by either replacing entries that cover the specific source phrase, or adding the alternative phrase translations to it, so that they are in competition.

This problem has only recently started to be explored in NMT and, in most of the cases, the proposed solutions integrate external knowledge at training stage. Time-consuming training routines, however, limit the suitability of this strategy for applications requiring real-time translations. In Gulcehre et al. (2015), monolingual data is used to train a neural language model that is integrated in the NMT decoder by concatenating their hidden states. In Arthur et al. (2016), the probability of the next target word in the NMT decoder is biased by using lexicon probabilities computed from a bilingual lexicon. When the external knowledge is in the form of linguistic information, such as POS tags or lemmas, Sennrich and Haddow (2016) propose to compute separate embedding vectors for each linguistic information and then concatenate them, without altering the decoder. Other solutions exploit the strengths of PBSMT systems to improve NMT by pre-translating the source sentence. In Niehues et al. (2016), the NMT model is fed with a concatenation of the source and its PBSMT translation. Some of these solutions lead to improvements in performance, but they all require time-intensive training of the NMT models to use an enriched input representation or to optimize the parameters of the model. (Stahlberg et al., 2016) proposed an approach which can be used at decoding time. A hierarchical PBSMT system is used to generate the translation lattices, which are then re-scored by the NMT decoder. During decoding, the NMT posterior probabilities are adjusted using the posterior scores computed by the hierarchical model. However, by representing the additional information as a translation lattice, this approach does not allow the use of external knowledge in the form of bilingual terms or quality judgements as we do in §5 and §6. A different technique is post-processing the translated sentences. Jean et al. (2015) and Luong and Manning (2015) replace the unknown words either with the most likely aligned source word or with the translation determined by another word alignment model.

The closest approach to ours is the one by (Hokamp and Liu, 2017). They explore all the possible constraints (or translation options) at each time step making sure not to generate a constraint that have already been generated in the previous timestep. Their approach generates all the constraints in the final output, thus implicitly it assumes that only one translation options is provided as constraint for a given source word/phrase. However, in a more realistic scenario (*e.g.* in presence of a termbase or when the target language is more inflected than the source language), a source word can have multiple translation options from which the decoder should decide the best one on-the-fly depending on the source context. Our approach can handle both scenarios thus being more suitable in practice. In this paper we consider the possibility of having multiple translation options for a single word. For this reason, we can not compare the guided decoder against the approach proposed in (Hokamp and Liu, 2017). In addition to the application in MT to customize NMT output to meet customer-specific needs (Task 1), our approach can also be used to add quality judgements within NMT at decoding time (Task 2).

3 NMT decoding

In this section we first provide a general introduction to NMT as it is currently commonly implemented in systems like the one used in our experiments. Then, we discuss its limitations with respect to our problem: guiding decoding with external knowledge.

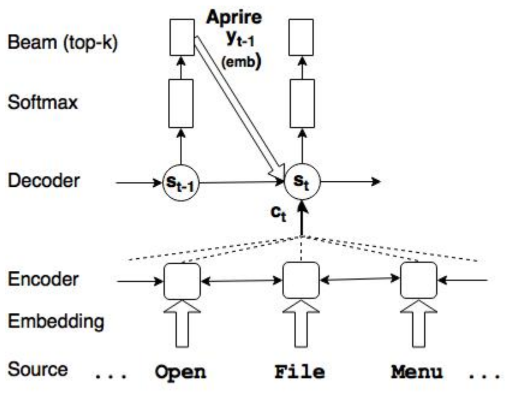

Figure 1: Overview of NMT decoding.

As shown in Figure 1, NMT starts by mapping all words of the source sentence into a continuous space, through an embedding layer. The word embeddings are processed by a bidirectional recurrent layer, implemented with gated recurrent units (GRUs) (Cho et al., 2014), which encodes each source word together with its left and right context in a sequence of hidden states. Once all the hidden states of the encoder are computed, the decoder starts generating the target string one word at the time. The decoder layer is implemented as a unidirectional GRU-based RNN.[1] The decoder hidden state at time t (s_t) is recursively updated via the previous hidden state (s_{t-1}), the embedding of the previously generated target word (y_{t-1}), and the input context (c_t). The context is generated as a convex combination of the encoder hidden states, whose weights ($\alpha_{t,j}$) are computed by a so-called attention model (Bahdanau et al., 2014). The attention model weights $\alpha_{t,j}$ are computed with a feed-forward neural network and can be interpreted as probability distributions over the source positions ($j = 1, \dots, m$).

Finally, the decoder linearly combines the embedding of y_{t-1}, s_t, and c_t and applies a *softmax* transformation to compute a language model over the target vocabulary which, through s_t, is actually conditioned on all the previous target words $y_0, \dots, y_{t-1}$ (where y_0 is a conventional sentence delimiter symbol). This language model is used to sample the new target word y_t, which is fed back to the decoder layer to continue the process of generating target words until the sentence delimiter word is produced. When a beam search strategy is employed, the most probable K target words ($y_{t,i}\ i = 1, \dots, K$) are sampled instead, and used as alternative hypotheses for the next decoding step. The process does not diverge because only K best target words are again selected from the resulting K target language model distributions. Through simple bookkeeping, the best target word sequence is computed that maximises the product of all the corresponding language model scores.

In this NMT workflow, there is no easy way to integrate partial translations provided by an external resource, such as a bilingual dictionary. Differently from a PBSMT decoder that is aware of which source phrase is translated at each step, the NMT decoder does not have this information. The only indirect connection between the target word y_t generated at time t and the corresponding source positions (words) is represented by the attention model weights $\alpha_{t,j}$, which are used to create the context vector c_t from the encoder hidden states. Moreover, differently from decoding in PBSMT, the NMT architecture described above does not apply any coverage constraint on the source positions. Thus, there is no guarantee that the output generated by NMT covers (*i.e.* translates) each source word exactly once.

4 Guided NMT decoding

To overcome the aforementioned problems, we present a novel technique called "guided decoding" that forces the decoder to generate particular translations given as external knowledge. Translation hints are provided in the form of annotations of individual source words in the input text. Our decoder accepts input in an XML format similar to the one adopted in the Moses toolkit,[2] which contains the source sentence and its annotations

[1]The implementation in Nematus (Sennrich et al., 2016a) we are building on deploys two GRUs, but this variation does not play any important role here.

[2]goo.gl/ObB6QL

as shown below (for English-German):
<seg id="1702"> enter the <n translation="Benutzer"> user</n> name and password </seg>
The annotations are placed in a "*n*" tag that has the attribute "*translation*" to hold the translation recommendation for the corresponding source word. The decoder parses the XML input and creates two parallel input streams: one that contains source words and another that contains the corresponding suggestions or the empty string. Then, the overall process is carried out similarly to the previously described NMT system but with a different interaction between the beam search and the network. In particular, after a new beam of K-top target words is generated, the "guide" mechanism checks the K hypotheses and their attention model weights to possibly influence the beam search with the external suggestions. This is done by: *i)* prioritizing the hypotheses that can generate the suggestions provided (§4.1); *ii)* performing look-ahead steps with the beam search to evaluate the current hypothesis (§4.2) and *iii)* applying different strategies to manage out-of-vocabulary (OOV) terms (§4.3).

4.1 Forcing the presence of a given term

In PBSMT, XML markups can be easily handled: when looking for translation options for each source phrase, the decoder checks both the external suggestions and the options in the phrase table. However, the NMT process is too complex to follow a similar approach. When generating a target word, NMT assumes a continuous representation of the whole source sentence through a context vector. In particular: *i)* all the source words can in principle contribute to generate a target word, and *ii)* different hypotheses may focus on different source words in the same decoding step. Thus, it is not guaranteed that the output at a given time step is solely dependent on a particular source word and, in turn, it is not clear how the external suggestions could be used. We tackle this issue by using the probability distribution of the source positions obtained from the attention model used to create the context vector. At each step of the beam search, for each of the K generated target words we look for the most probable source position provided by the attention model. If the corresponding source word has a suggestion, then we replace the target word by the given suggestion and update the score of the hypotheses; otherwise, we keep the original target word.

4.2 Placing the term in the right position

The guiding mechanism in §4.1 allows the decoder to generate a given translation by replacing options inside the beam. However, the method does not consider cases in which one source word position is involved in the generation of multiple target words. This may happen when the decoder has its attention on a particular source word more than once (*e.g.* an article and a noun in the target referring to the same noun in the source). In these situations, it could happen that valid translation options are erroneously replaced and the external suggestion is reproduced multiple times in the output. For instance, in Figure 2, the source word "$application$" which the attention model refers to for both "die" and "$Anwendung$" would be translated as "$Anwendung\ Anwendung$".

To address this problem and to make our approach more robust to possible attention model nuances, we relax the hard replacement of a translation option if it differs from the provided suggestion. In particular, if the conditions for a replacement occur, we also check if the beam search would nevertheless generate the suggestion from the current word, within a small number of steps. If this happens, we keep the current word in place since we know that the actual suggestion will be generated in the short future. If the suggestion is not reachable, then we force the replacement.

Algorithm 1 illustrates the modified beam search process that generates the K best hypotheses for the next target word. Starting from the beam at time $t-1$, a new state S_t is computed and returned. The state contains the best K target words (y_t), their corresponding decoder hidden states (s_t), cumulative language model scores (q_t), backtracking indexes to the parent entries in the previous state (b_t), and source indexes having the largest attention weight (α_t). In addition, the modified beam search algorithm maintains, for each of the K entries, the list of suggestions (L_t) that have been generated within that hypothesis so far. The algorithm accesses the global variable $\tilde{y}[j]$, which contains for each source position j either a provided target word suggestion or the empty word $\emptyset$. The algorithm proceeds by computing the normal beam search step (line 14) and initializing the lists of generated suggestions with the list of the

Algorithm 1 Guided Beam Search Step

```
1: ▷ $K$: size of beam
2: ▷ $L_t$: K lists of generated suggestions
3: ▷ $N$: look-ahead step to check reachability
4: ▷ $S_t$ = $[y_t, s_t, q_t, b_t, \alpha_t]$: state information
5: ▷ $y_t$: K target words
6: ▷ $s_t$: K decoder layer hidden states
7: ▷ $q_t$: K cumulative language model scores
8: ▷ $b_t$: K backtracking indexes
9: ▷ $\alpha_t$: K highest-attention-indexes
10: ▷ Global variable with suggestions:
11: ▷   $\tilde{y}[j]$: target word for source position $j$
12: procedure GUIDEDBEAMSEARCH($K$,L$_{t-1}$,$N$,S$_{t-1}$)
13:     ▷ Perform a step of beam search
14:     $S_t$:= BeamSearchStep($S_{t-1}$)
15:     ▷ Copy generated suggestions from parent
16:     $L_t$:=UpdateLists($b_t, L_{t-1}$)
17:     ▷ for each entry of the beam
18:     for $k \in \{1, \ldots, K\}$ do
19:         ▷ Check suggestion for source word $\alpha_{t,k}$
20:         if $\tilde{y}[\alpha_{t,k}] \neq \emptyset \wedge \alpha_{t,k} \notin L_{t,k}$ then
21:             $\tilde{y} := \tilde{y}[\alpha_{t,k}]$
22:             if $y_{t,k} \neq \tilde{y}$ then
23:                 ▷ if $\tilde{y}$ is not generated by $N$ steps
24:                 if !Reachable($S_t, \tilde{y}, k, N$) then
25:                     ▷ Force suggestion in beam
26:                     $y_{t,k} = \tilde{y}$;
27:                     ▷ Update suggestion list
28:                     Add($\alpha_{t,k}, L_{t,k}$)
29:                 end if
30:             else
31:                 ▷ Suggestion is generated
32:                 Add($\alpha_{t,k}, L_{t,k}$)
33:             end if
34:         end if
35:     end for
36:     return ($L_t, S_t$)
37: end procedure
```

corresponding parents (line 16) that are accessible through the backtracking indexes. The main loop (line 18) checks, for each beam entry, the source position that received the highest weight by the corresponding attention model. If this source position ($\alpha_{t'k}$) corresponds to a non-empty suggestion and if the suggestion has not been generated by one of the predecessors of this entry, then the algorithm decides whether or not this suggestion ($\tilde{y}$) has to be forced in the beam. In particular, there are two cases for which action is taken. First, if the suggestion is different from the word in the beam (line 22) and the suggestion will not be generated by one of its next N successors, then the suggestion will replace the current word (line 26). The list of generated suggestions by this hypothesis is updated accordingly. Second, if the suggestion is equal to the word in the beam (line 30), then the suggestion has been generated directly by the beam search and the corresponding list is updated (line 32). The algorithm finally returns the updated lists of generated suggestions and the updated beam search state.

This algorithm can generate both continuous and discontinuous target phrases:

Continuous phrases are those in which consecutive target words are pointed by the same source word. The phrase pair ("*application*", "*die Anwendung*") in Figure 2 falls in this category. With a look-ahead window set to 1 in the algorithm, the decoder will be able to generate bigram phrases (such as "*die Anwendung*"). With larger look-ahead windows, longer phrases can be generated.

Discontinuous phrases are those in which target words pointed by the same source word are intermingled with other words for which the attention points elsewhere. The phrase pair ("*quit*", "*haben verlassen*") in Figure 2 falls in this category. In these cases, the guided beam search should look at least two steps in the future. The time step value maps the distance (number of words) between the left and right sub-parts of the target phrase. In our example, the distance is 4 (*i.e.* 4 steps are needed to reach "*verlassen*" from "*haben*") so, if we set the look-ahead window to 4, the decoder can generate the annotation "*verlassen*" after emitting "*haben*".

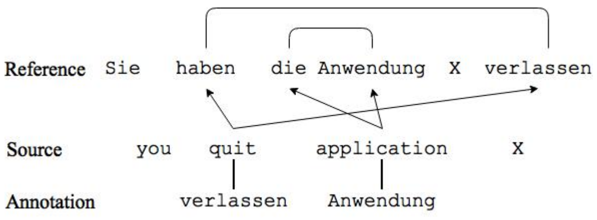

Figure 2: An example showing continuous ("*die Anwendung*") and discontinuous ("*haben...verlassen*") target phrases.

4.3 Guiding the translation of OOV terms

The last problem is dealing with suggestions that are OOV words. In NMT, it is common practice to replace OOV words by the unknown token (UNK) and use its corresponding embedding. The questions are: *i)* if an OOV suggestion is given by the external resource, should the modified beam search force it into the beam?, and *ii)* which target word embedding should be used in the next step? To answer these questions, we implemented a lookup table to store all the OOV suggestions along with their unique *id* before initializing the decoder. These *id*s are used for OOV suggestions by the beam search instead of the *id* associated by default to the UNK token. To get the embeddings for OOV suggestions, we tested different strategies, which are discussed in §5.2 and §6.2.

5 Task 1: Machine Translation

In our first experiment, we use guided decoding in a standard MT setting. Our goal is to improve MT performance by exploiting prior knowledge supplied as translation recommendations for domain-specific terms. The suggested terms (*i.e.* the constraints posed to the decoder) are usually few, thus leaving a large degree of freedom to the NMT decoder while generating the output.

5.1 Experimental setting

NMT models. We evaluate guided decoding in its ability to improve the performance of two different English to German NMT models, both obtained with the Nematus toolkit (Sennrich et al., 2016a). The first system operates at word level and it is trained by using part of the JRC-Acquis corpus (Steinberger et al., 2006), Europarl (Koehn, 2005) and OpenSubtitles2013 (Tiedemann, 2009), which results in at total of about 1.8M parallel sentence pairs. The size of the vocabulary, word embedding, and hidden units is respectively set to 40K, 600, and 600, and parameters are optimised with Adagrad (Duchi et al., 2011) using a learning rate of 0.01. The batch size is set to 100, and the model is trained for 300K updates ($\sim$17 epochs). At test stage, the word-level system is supplied with terminology lists containing term recommendations at the level of granularity of full words. The second system is trained on sub-word units by using the Byte-Pair Encoding (BPE) technique (Gage, 1994), which has been proposed by Sennrich et al. (2016b) as a successful way to reduce the OOV rate. The system used in our evaluation is the pre-trained model built for the best English-German submission (Sennrich et al., 2016a) at the News Translation task at WMT'16 (Bojar et al., 2016). At test stage, it is supplied with terminology lists containing term recommendations in BPE format. In all the experiments we use a default beam size of 12.

Test data. We experiment with two domain-specific English-German test sets containing 850 segments each: *i)* a subset of the EMEA corpus (Tiedemann, 2009) for the medical domain and *ii)* an information technology corpus extracted from software manuals (Federico et al., 2014). Word-level term lists for both domains are obtained by processing the test data with the "Terminology as a Service" platform,[3] a cloud-based system that supports automatic bilingual term extraction from user uploaded documents. The BPE-level version of each word-level term list is obtained as follows. First, each entry is segmented with the BPE rules available along with the pre-trained Nematus model. Then, the segmented entries are aligned by running MGiza++ (Gao and Vogel, 2008) trained on the BPE-level WMT'16 training data. Finally, all the one-to-one aligned sub-units are extracted to form the sub-word level bilingual term dictionaries. The word and sub-word bilingual dictionaries are used to annotate the respective test sets. This results in the annotation of $\sim$5K words and $\sim$7.5K sub-words. The term rate (#Term/#Tokens) in the two test sets is respectively 18.9% (IT: 21.5% and Medical: 17.7%) and 20.1% (IT: 23.2% and Medical: 18.0%).

5.2 Results and discussion

Our results on the MT task are reported in Table 1, which shows system performance on the concatenation of the test sets from the two target domains. Performance is measured with BLEU (Papineni et al., 2002), and statistical significance is computed with bootstrap resampling (Koehn, 2004). The result of the word-level baseline system is computed after post-processing its output following the approach of Jean et al. (2015), which was customized to our scenario. This method (see §2) is driven by the attention model to replace the UNK tokens in the output with their corresponding recommendation supplied as external knowledge. This post-processing strategy is not used for the BPE-level baseline because it implicitly addresses the problem of OOVs.

We evaluate our guided decoder incrementally, by adding one at a time the mechanisms described in §4. In the discussion, we do not compare the performance of BPE-based and word-based models because the former were trained and optimized with larger training data for the news translation task at WMT'16. Results, instead, will be discussed in terms of the contribution yield by each mechanism on top of previous best results.

[3]`http://www.taas-project.eu`

The baseline decoder (Baseline) performs better than our basic guided decoder (GDec_base), which considers translation recommendations only in the case of known terms as described in §4.1. This indicates that the problem of constraining the NMT output using a bilingual dictionary can not be addressed by simply emitting the recommendations whenever the corresponding source term has the highest attention.

	words	BPE
Baseline	22.62	25.64
GDec_base	21.68	25.25
GDec_base+oov	23.04†	25.66
GDec_base+oov+reach	**25.51†**	**28.42†**

Table 1: BLEU results of different decoders on the MT task ("†" indicates statistically significant differences wrt. Baseline with p<0.05).

GDec_base+oov extends GDec_base with the mechanism to handle OOV annotations as described in §4.3. In order to generate word embeddings for OOV terms, we tested several strategies: *i)* using the embedding of the unknown word, *ii)* using the embedding of the best target word in the beam, *iii)* using the embedding of the previous word (y_{t-1}), and *iv)* using the average of the embeddings of all the previous words ($y_{1,..,t-1}$). The best results are obtained when using the embedding of the unknown word which, on further investigation, resulted to be close to rare words in terms of cosine similarity. As of now, this proximity to rare words suggests that it can model OOVs better than the other strategies, but deeper investigations on this aspect are certainly an interesting topic for future analysis. The ability to handle OOVs yields statistically significant improvements (+0.4 BLEU) over the baseline for the word-based model. In contrast, since the BPE-based systems can implicitly mitigate the OOV problem (as discussed in §5.1), our strategy results in marginal improvements over the BPE baseline.

Finally, GDec_base+oov+reach combines OOV handling with the method to avoid repetitions and to manage the insertion positions described in §4.2. Since it uses a look-ahead (LA) hyper-parameter in order to validate the translation options, we experimented with different values ranging from 1 to 9. By varying the LA window, performance increases both for the word-level and the BPE-level models up to the highest scores achieved with LA=6. Increasing LA beyond 6 does not yield further gains. Using LA=1 performs slightly worse (-0.4 BLEU) than LA=6, indicating that this mechanism is already effective even at small values (*i.e.* on our data, a large number of problematic cases involve continuous phrases like the example in Figure 2). This full-fledged guided decoder achieves a statistically significant improvement of $\sim$3 BLEU points over both word-level and BPE-level baselines. A per-domain results' analysis shows similar gains over the Baseline for word-based (IT: +4.3, Medical: +2.6) and for BPE-based NMT (IT: +3.9, Medical: +2.0).

To better understand the behaviour of our decoder, we further analysed its output. First, the percentage of translation recommendations produced in the MT output ($\frac{\#TermsInTranslation}{\#AnnotatedTerms}$) was computed both for the Baseline, and for the GDec_base+oov+reach decoders. As expected, the Baseline achieves lower results (BPE-level: 70.81%, Word-level: 65.38%) compared to the full-fledged GDec_base+oov+reach (BPE-level: 94.08%, Word-level: 87.19%). Indeed, as a generic NMT system, it is not able to properly handle domain-specific terms (the BPE representation helps to reduce OOVs but does not guarantee correct realizations in the target language). Second, a preliminary error analysis was carried out by looking at the word alignments returned by the attention model. This revealed that the majority of the errors produced by our decoder can be found in sentences in which the annotated source words never receive the highest attention, thus making the corresponding recommendations unreachable.

Manual Analysis: We manually analyzed some samples to understand the effect of using GDec in the translation task. We observe that when fed with a list of translation options in the form of xml annotations, GDec is able to generate the correct terms. Moreover, these local improvements also help GDec to fix other parts of the translation. Examples illustrating these effects are provided in Table 2. Example 1 shows that the baseline system (Base) translates the word "browse" in the source sentence (Src) to "stöbern" (En: "rummage around") but the post-editors prefers to use in the reference "durchsuche" (En: "search") which is then generated by the GDec. Example 2 illustrates a case where post-editors prefer to preserve the terms in the source language rather than translating them. The baseline system translates "plastics labs" to "Kunststofflabore" (En: "Plastic laboratories"), however, it should be preserved *as-is* in the final output as done by GDec. Another interesting example highlighting the effect of choosing a correct term on the overall translation quality is provided in example 3. The term "zugewiesen" translation of the source word "assigned" helps GDec to correct other parts of the final translation, like generating "es gibt keine" (En: "There is no") which is otherwise missing in the baseline translation.

6 Task 2: Automatic Post-Editing

In our second experiment, we apply guided decoding in an automatic post-editing task. The goal of automatic post-editing (APE) is to correct errors in an MT-ed text. The problem is typically approached as a "monolingual translation" task, in which models are trained on parallel corpora containing (*MT_output*, *MT_post-edit*) pairs, with MT post-edits coming from humans (Simard et al., 2007; Chatterjee et al., 2015b, 2017). In their attempt to translate the entire input sentence, APE systems usually tend to over-correct the source words, *i.e.* to use all applicable correction options. This can happen even when the input is correct, often resulting in text deterioration (Bojar et al., 2015). To cope with this problem, neural-based APE decoders would benefit from external knowledge indicating words in the input which are correct and thus should not be modified during decoding. For that we propose to use word-level binary quality estimation labels (Blatz et al., 2004; de Souza et al., 2014) to annotate the "good" words that should be kept. Due to the relatively high quality of the MT outputs (62.11 BLEU), source sentences will usually contain many terms annotated as "good". This, compared to the MT task, poses more constraints on the decoder.

6.1 Experimental setting

NMT models. We use the pre-trained model built for the best English-German submission (Junczys-Dowmunt and Grundkiewicz, 2016) at the WMT'16 APE task. This available model was trained with Nematus over a data set of ∼4M back-translated pairs, and then adapted to the task-specific data segmented using the BPE technique.

Test data. In this experiment, we use the English-German data released at the WMT'16 APE shared task (Bojar et al., 2016). To annotate the test set, instead of relying on automatic quality, predictions, we exploit oracle labels indicating "good" words (to be kept in the output) and "bad" words (to be replaced by the decoder). To this aim, we first aligned each MT output with the corresponding human post-edit using TER (Snover et al., 2006). Then, each MT word that was aligned with itself in the post-edit was annotated as "good". This resulted in a high number of "good" labels (on average, 79.4% of the sentence terms). It is worth noting that, by construction, the resulting quality labels are "gold" annotations that current word-level quality estimation systems can only approximate. These make them suitable for our testing purposes, as they allow us to avoid the noise introduced by sub-optimal predictors. The BPE-level version of the test set is obtained by projecting the word-level QE tags into the sub-words (all sub-words of a word receive the original word tag). If a sub-word was labelled as "good", then we annotate it with itself to indicate that the decoder must generate the sub-word in the output.

6.2 Results and discussion

Our results on the APE task are reported in Table 3. Performance is measured with the two WMT'16 APE task metrics, namely TER and BLEU (Bojar et al., 2016). The statistical significance for BLEU is computed using paired bootstrap resampling, while for TER we use stratified approximate randomization (Yeh, 2000).

Our first baseline (`Base-MT`), the same used at WMT, corresponds to the original MT output left untouched. Our second baseline (`Base-APE`) is a neural APE system that was trained on (*MT_output*, *MT_post-edit*) pairs but ignores the information from the QE annotations. `Base-APE` improves the `Base-MT` up to 3.14 BLEU points.

Similar to §5.2, the evaluation of our guided decoder is performed incrementally. `GDec_base` forces the "good" words in the automatic translation to appear in the output according to the mechanism described in §4.1. This basic guidance mechanism yields only marginal improvements over the `Base-MT` and is far behind the `Base-APE`. This can be explained by the large number of constraints (*i.e.* "good" words to be

Src: <n translation="durchsuchen||Durchsuchen"> browse </n> all products
Base: stöbern alle Produkte
GDec: durchsuchen Sie alle Produkte
Ref: durchsuchen Sie alle Produkte

Src: <n translation="produkt||Produkt"> Product </n> 4 - <n translation="plastics||Plastics"> Plastics </n> <n translation="labs||Labs"> Labs </n>
Base: Produkt 4 - Kunststofflabore
GDec: Produkt 4 - Plastics Labs
Ref: Produkt 4 - Plastics Labs

Src: There is no limit on the number of valve gates that can be <n translation="zugewiesen"> assigned </n> to a model .
Base: die Anzahl der Ventiltore , die einem Modell zugeordnet werden können , ist nicht begrenzt .
GDec: es gibt keine Grenze für die Anzahl der Ventiltore , die einem Modell zugewiesen werden können .
Ref: Es gibt keine Begrenzung für die Anzahl der Anschnitte , die zugewiesen werden können.

Table 2: Examples covering some cases where GDec improves over the baseline for the MT task

	BLEU (↑)	TER (↓)
`Base-MT`	62.11	24.76
`Base-APE`	65.25	23.67
`GDec_base`	62.68†	23.97†
`GDec_base+OOV`	62.69†	23.96†
`GDec_base+OOV+reach`	**67.03**†	**22.45**†

Table 3: Performance of different decoders on the APE task measured in terms of TER (↓) and BLEU score (↑) ("†" indicates statistically significant differences wrt. `Base-APE` with p<0.05).

kept), which drastically reduces the freedom of the decoder to generate surrounding words. This is confirmed by manual inspection: many original MT segments were missing function words that depended on the "good" words present in the sentence. These insertions are easily performed by the unconstrained `Base-APE` decoder but are unreachable by `GDec_base`, which is only able to keep the annotated words.

`GDec_base+OOV` integrates the mechanism to handle OOV annotations described in §4.3. Since the model is trained on the BPE segment corpus, the problem of OOV is already tackled by the model itself. Thus, we do not observe a significant contribution by this mechanism, which is in-line with our results on BPE in the MT task.

`GDec_base+OOV+reach` is our full-fledged system, which manages repetitions and insertion positions as illustrated in §4.2. Its ability to better model the surroundings of the annotated words allows this technique to achieve statistically significant improvements (+1.78 BLEU, -1.22 TER) over the strong `Base-APE` decoder.

To better appreciate the ability of the APE decoder to leverage the QE labels and to avoid over-correction, we compute the APE precision (Chatterjee et al., 2015a) as the ratio of the number of sentences an APE system improves (with respect to the MT output) over all the sentences it modifies. The `GDec_base+OOV+reach` decoder gains 9 precision points over `Base-APE` (72% vs. 63%) confirming that guided decoding supported by QE labels can improve also APE output quality.

Manual Analysis: Similar to the MT task we performed a manual analysis of the outputs generated by different APE systems. Examples capturing various aspects of the workings of GDec in this task are provided in Table 4. The labels Src, MT, Base, GDec, and Ref respectively represents the source sentence, machine translation output, baseline APE output, GDec full-fledge output, and the reference translation. Example 1 shows the capability of GDec to preserve the MT words in the final output that are correctly generated by the MT system. In this example the word "Gibt" (En: "Specifies") is preserved by GDec which is otherwise translated to "Legt" (En: "Sets") by the baseline system. Example 2 shows that guiding the neural decoder by marking the MT word "gewährleisten" (En: "ensure") as "Good" not only helps to preserve it in the final output but also help to improve other parts of the translation like "um ein ähnliches" (En: "a similar") which is otherwise untouched by the baseline APE system.

Src: Specifies the source for the glow .
MT: <n translation="gibt||Gibt">Gibt</n> die Quelle</n> für</n> das Glü@@ hen aus .
Base: Legt die Quelle für das Glühen fest .
GDec: Gibt die Quelle für das Glühen aus .
Ref: Gibt die Quelle für den Schein an .

Src: Map Japanese indirect fonts across platforms to ensure a similar appearance .
MT: ... zu einem ähnlichen Erscheinungsbild <n translation="gewährleisten"> gewährleisten </n> .
Base: ... " auf einem ähnlichen Erscheinungsbild an .
GDec: ... " auf , um ein ähnliches Erscheinungsbild zu gewährleisten .
Ref: ... zu , um ein ähnliches Erscheinungsbild zu gewährleisten .

Src: All values , even primitive values , are objects .
MT: alle Werte , auch Grund@@ werte , <n translation="handelt">handelt</n> <n translation="es">es</n> <n translation="sich">sich</n> <n translation="um">um</n> <n translation="Objekte">Objekte</n> .
Base: Alle Werte , auch Grundwerte , sind Objekte .
GDec: Alle Werte , auch Grundwerte , handelt es sich um Objekte .
Ref: Bei allen Werten , auch Grundwerten , handelt es sich um Objekte .

Table 4: Examples covering some cases where GDec improves over the baseline for APE task.

Example 3 illustrates that GDec can be very useful to avoid the problem of over-correction. The MT segment in this example is almost a correct translation of the source sentence and should be left untouched but the baseline APE system modifies it deteriorating the overall translation quality. However, when the MT word is annotated to itself by the xml tags, GDec is able to preserve this word thereby avoiding over-correction and retaining the translation quality.

7 Conclusion

We presented a novel method for guiding the behaviour of an NMT decoder with external knowledge supplied in the form of translation recommendations (*e.g.* terminology lists). Our approach supervises the translation process, ensuring that the final output includes the expected translations, in the right place, including cases of added OOV words. Evaluation results on two tasks indicate the effectiveness of our proposed solution, which significantly improves over a standard NMT decoder.

Acknowledgments

This work has been partially supported by the EC-funded H2020 projects QT21 (grant agreement no. 645452) and ModernMT (grant agreement no. 645487).

References

Philip Arthur, Graham Neubig, and Satoshi Nakamura. 2016. Incorporating Discrete Translation Lexicons into Neural Machine Translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. "neural machine translation by jointly learning to align and translate". *arXiv preprint arXiv:1409.0473* .

John Blatz, Erin Fitzgerald, George Foster, Simona Gandrabur, Cyril Goutte, Alex Kulesza, Alberto Sanchis, and Nicola Ueffing. 2004. Confidence Estimation for Machine Translation. In *Proceedings of the 20th International Conference on Computational Linguistics*.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016. Findings of the 2016 Conference on Machine Translation. In *Proceedings of the First Conference on Machine Translation*. Association for Computational Linguistics, Berlin, Germany.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. 2015. Findings of the 2015 Workshop on Statistical Machine Translation.

In *Proceedings of the Tenth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, Lisbon, Portugal.

Rajen Chatterjee, Gebremedhen Gebremelak, Matteo Negri, and Marco Turchi. 2017. Online Automatic Post-editing for MT in a Multi-Domain Translation Environment. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*. Valencia, Spain, pages 525–535.

Rajen Chatterjee, Marco Turchi, and Matteo Negri. 2015a. The FBK Participation in the WMT15 Automatic Post-editing Shared Task. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*. Lisbon, Portugal.

Rajen Chatterjee, Marion Weller, Matteo Negri, and Marco Turchi. 2015b. Exploring the Planet of the APEs: a Comparative Study of State-of-the-art Methods for MT Automatic Post-Editing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*. Beijing, China.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. In *Proceedings of SSST@EMNLP 2014, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Doha, Qatar.

José G. C. de Souza, Jesús González-Rubio, Christian Buck, Marco Turchi, and Matteo Negri. 2014. FBK-UPV-UEdin participation in the WMT14 Quality Estimation shared-task. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*. Baltimore, MD, USA.

John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research* .

Marcello Federico, Nicola Bertoldi, Mauro Cettolo, Matteo Negri, Marco Turchi, Marco Trombetti, Alessandro Cattelan, Antonio Farina, Domenico Lupinetti, Andrea Martines, Alberto Massidda, Holger Schwenk, Loïc Barrault, Frederic Blain, Philipp Koehn, Christian Buck, and Ulrich Germann. 2014. THE MATECAT TOOL. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: System Demonstrations*. Dublin, Ireland.

Philip Gage. 1994. A New Algorithm for Data Compression. *C Users Journal* 12(2).

Qin Gao and Stephan Vogel. 2008. Parallel Implementations of Word Alignment Tool. In *Software Engineering, Testing, and Quality Assurance for Natural Language Processing*.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loïc Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On Using Monolingual Corpora in Neural Machine Translation. *arXiv e-prints* .

Chris Hokamp and Qun Liu. 2017. Lexically Constrained Decoding for Sequence Generation Using Grid Beam Search. *CoRR* abs/1704.07138.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On Using Very Large Target Vocabulary for Neural Machine Translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. Beijing, China.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2016. Log-linear Combinations of Monolingual and Bilingual Neural Machine Translation Models for Automatic Post-Editing. In *Proceedings of the First Conference on Machine Translation*. Berlin, Germany.

Philipp Koehn. 2004. Statistical Significance Tests for Machine Translation Evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Conference Proceedings: the tenth Machine Translation Summit*. AAMT, Phuket, Thailand.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*.

Minh-Thang Luong and Christopher D Manning. 2015. Stanford Neural Machine Translation Systems for Spoken Language Domains. In *Proceedings of the International Workshop on Spoken Language Translation*.

Jan Niehues, Eunah Cho, Thanh-Le Ha, and Alex Waibel. 2016. Pre-Translation for Neural Machine Translation. In *International Conference on Computational Linguistics*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*.

Rico Sennrich and Barry Haddow. 2016. Linguistic Input Features Improve Neural Machine Translation. In *Proceedings of the First Conference on Machine Translation*. Berlin, Germany.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Edinburgh Neural Machine Translation Systems for WMT 16. In *Proceedings of the First Conference on Machine Translation*. Berlin, Germany.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Michel Simard, Cyril Goutte, and Pierre Isabelle. 2007. Statistical Phrase-Based Post-Editing. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT)*. Rochester, New York, pages 508–515.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of Association for Machine Translation in the Americas*. Cambridge, Massachusetts, USA, pages 223–231.

Felix Stahlberg, Eva Hasler, Aurelien Waite, and Bill Byrne. 2016. Syntactically Guided Neural Machine Translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Berlin, Germany.

Ralf Steinberger, Bruno Pouliquen, Anna Widiger, Camelia Ignat, Tomaz Erjavec, Dan Tufis, and Dániel Varga. 2006. The JRC-Acquis: A Multilingual Aligned Parallel Corpus with 20+ Languages. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC'2006)*. Genoa, Italy.

Jörg Tiedemann. 2009. News from OPUS - A Collection of Multilingual Parallel Corpora with Tools and Interfaces. In *Recent Advances in Natural Language Processing*, John Benjamins, Amsterdam/Philadelphia, Borovets, Bulgaria.

Alexander Yeh. 2000. More Accurate Tests for the Statistical Significance of Result Differences. In *Proceedings of the 18th Conference on Computational Linguistics - Volume 2*.