Universität Zürich^UZH

Bachelor's Thesis
presented to the Faculty of Arts and Social Sciences
of the University of Zurich
for the degree of
Bachelor of Arts

Lexically Constrained Decoding using Insertion Transformer

Author: Tim Graf
Student ID: 15-737-083

Examiner: Prof. Dr. Rico Sennrich

Department of Computational Linguistics

Submission Date: (01.12.2020)

Abstract

This thesis suggests the use of the Insertion Transformer [Emelianenko et al., 2019] to ameliorate lexically constrained decoding in neural machine translation. Previous works have had issues with guaranteeing that a constraint appears in the translation while also ensuring that the term is properly inflected to match the rest of the sentence. My contribution is a decoding algorithm for the Insertion Transformer that achieves a guarantee that the constraint appears in the translation and allows morphological variation to be taken into account. Additionally, I show that this algorithm can also provide help with resolving conflicting terminology database entries. Although my experiments fail to reproduce state of the art machine translation quality, my experiments show that morphological variation can improve translation quality in cases where inflection of a constraint is needed.

Zusammenfassung

Diese Arbeit schlägt vor, den Insertion Transformer von Emelianenko et al. [2019] zu verwenden, um das sogenannte Lexically Constrained Decoding zu verbessern. Frühere Publikationen hatten Probleme damit, eine Garantie zu erreichen, dass ein Constraint tatsächlich in der Übersetzung vorkommt und gleichzeitig auch korrekt flektiert ist im Kontext des ganzen Satzes. Mein Beitrag ist ein Decoding-Algorithmus für den Insertion Transformer, womit eine Garantie erreicht werden kann, dass ein Constraint in der Übersetzung vorkommt und dem Modell trotzdem erlaubt, verschiedene morphologische Varianten des Constraints zu berücksichtigen. Zusätzlich zeige ich, dass dieser Algorithmus dabei helfen kann, Konflikte bei widersprüchlichen Einträgen in einer Terminologiedatenbank aufzulösen. Obwohl es mir in meinen Experimenten nicht gelingt, den ßtate of the artïn der maschinellen Übersetzung zu reproduzieren, zeigen meine Experimente, dass das Hinzufügen von morphologischer Variation die Übersetzungsqualität verbessern kann, wenn auch tatsächlich ein Constraint flektiert werden muss.

Acknowledgements

First an foremost, I want to thank my supervisor Rico Sennrich for his support over the course of this thesis. His constructive criticism and invaluable advice have played a critical role in shaping this thesis. Also, his kindness and patience have made it a pleasure to meet with him and engage in interesting and productive discussions.

I would also like to thank my family and friends for their support during this thesis, and especially my partner for her unconditional support during the stressful parts of my work on this thesis.

Contents

List of Figures

List of Tables

List of Acronyms

BLEU	Bilingual Evaluation Understudy
BPE	Byte Pair Encoding
CBS	Constrained Beam Search
DBA	Dynamic Beam Allocation
GBS	Grid Beam Search
GPU	Graphics Processing Unit
IATE	Interactive Terminology for Europe
INTRUS	Insertion Transformer for Unconstrained Order Sequence Modeling
MT	Machine Translation
NMT	Neural Machine Translation
SMOR	Stuttgarter Morphologisches Analysewerkzeug
SMT	Statistical Machine Translation
WMT	Workshop on Machine Translation
Zmorge	Zurich Morphological Analyzer for German

1 Introduction

1.1 Motivation

Neural machine translation (NMT) has yielded large improvements in translation quality compared to previous statistical machine translation (SMT) approaches. While the output of NMT systems is generally considered to be more fluent than their SMT counterparts, it is more difficult to enforce constraints on the target side of the translation. In an industry setting, such constraints are especially important as translators are often required to follow certain terminology guidelines, for example to maintain the Corporate Language of a customer. To ensure a terminologically correct and consistent translation, most actors in the translation industry maintain terminology databases which contain the relevant terminology for a specific domain. An entry in a terminology database usually consists of a term in the source and target language respectively, e.g. *ATM* (en) and *Bancomat* (de). Thus, if the word *ATM* appears in an English sentence that is to be translated into German, the translator is reminded that *ATM* should be translated as *Bancomat* (and not as *Bankautomat*, although it would be a valid translation), for example through highlighting in professional translation software. While this certainly aids the translator to produce terminologically consistent translations, the translation process can be made more efficient by ensuring that the output of an MT system already contains the necessary terms.

For phrase-bases SMT systems [Koehn et al., 2003], mechanisms to control the MT output have existed for quite some time, but because of the architectural change between SMT and NMT, these mechanisms could not simply be ported to NMT systems. However, in an effort to support similar functionality, there have been numerous approaches that have found a way to incorporate terminology into NMT, all of which have certain caveats.

A range of approaches uses forms of constrained decoding, where the terms are used as lexical constraints that have to be fulfilled in order for a hypothesis to be accepted [Hasler et al. [2018], Hokamp & Liu [2017], Post & Vilar [2018], Susanto et al. [2020]].

In these cases, the model must produce a translation where all the given constraints are met. While in theory this behaviour might be desirable, the hard constraints can also hurt translation quality in certain cases [Dinu et al., 2019]. Additionally, these methods lack the ability to inflect the terms, such that translators need to be cautious to find correct terms that appear in a incorrectly inflected form in the MT output. The work of Dinu et al. [2019] has shown that NMT systems are capable of inflecting terminology constraints, but their approach loses the guarantee that terms contained in the terminology database always appear in the translation.

Emelianenko et al. [2019] propose a novel way of generating the translation with an Insertion Transformer. Instead of generating the sentence in a left-to-right fashion, the NMT system can insert a new token at any position in the translation hypothesis. I suggest to use this capability of the Insertion Transformer to alleviate a frequent problem in conventional constrained decoding approaches, namely that lexical constraints often do not fit well into the translation. To solve the problem with missing inflections of terms, I generate morphological variations of a term, such that the model can choose which inflection of a term shall be used in a translation. With the ability to choose from different variations of a term, I also experiment to use the translation model to resolve conflicting entries in a terminology database, i.e. when a term has multiple plausible translations, as it is common in the IATE database.

Thus, the research questions that shall be answered in this thesis, are:

- How does using an Insertion Transformer compare to other methods for implementing lexical constraints on the output of NMT?
- Does the use of morphological variations in lexical constraints improve translation quality?
- Is using an Insertion Transformer beneficial when the terminology database contains noise, i.e. when there are multiple possible matches in the target language for a given source term?

1.2 Thesis Structure

In this first chapter I gave a brief introduction and provided the aims of this thesis. Chapter 2 introduces the necessary concepts needed to understand the remainder of this thesis, and gives an overview of the most important works that have solved parts of terminology integration in NMT. Chapter 3 covers the materials used for training and evaluation. In chapter 4, I will introduce my proposed algorithm for constrained decoding with an Insertion Transformer, as well as the methods I used to construct a meaningful testset. In chapter 5 I present the results of my experiments. Chapter 6 follows, where I provide an analysis of said results, before concluding my thesis in chapter 7.

2 Background

2.1 Limitations of Standard Beam Search

Current state-of-the-art systems in NMT typically use an encoder-decoder architecture [Sutskever et al., 2014]. In these frameworks, the encoder part of the neural network encodes a sentence into a vector-representation which is then decoded into a translation by the decoder part. The general underlying problem is that for a given source sentence, the model should produce the most probable translation. As computing the conditional probability for every possible translation is infeasible, the problem is simplified: The most probable translation is the most probable sequence of tokens, where the probability of the ith token is its conditional probability given the source sentence and all i-1 previously generated tokens. It is again not feasible to compute all possible sequences of tokens, thus the most probable sequence is approximated: always picking the optimal token at each timestep is prone to make a bad global choice, hence most systems approximate the most probable sequence through beam search [Sutskever et al., 2014]. With beam search, the k most probable hypotheses are kept at each timestep. In the next step, all possible continuations for each hypothesis are calculated and again, the k most probable of all new hypotheses are kept. In this example, k is then called the beam size. Beam search has a complexity of $O(|Y|Vk)$, where V is is the (target) vocabulary size and $|Y|$ the length of the translated segments in subwords.

With regards to the incorporation of terminology entries into beam search, the biggest underlying problem is that no hypothesis in the beam is guaranteed to include the desired translation of the term. As a result, the part of the search space covered by beam search needs to be increased and/or altered to achieve a high probability that the term is in one of the final hypotheses.

2.2 Constrained Beam Search

Some of the first approaches to incorporate terminology constraints into NMT included modifying the beam search algorithm in a way that the needed terms end up in the translation. Modifications to the decoding algorithm are convenient, as this does not require to retrain an NMT model, instead, a trained model can be used out of the box. Constrained beam search [CBS; Anderson et al. [2017]] showed that image captioning networks are able to decode with constraints. This approach, however, uses 2^C beams, where C is the number of constraints. As I intend to add morphological variations to the constraints, CBS certainly would not scale well in my case. Another proposed beam search modification was grid beam search [GBS; Hokamp & Liu [2017]]. GBS improves efficiency over CBS, because it only uses C beams instead of 2^C. Even though this is a considerate decrease in time complexity, this method is still no suited for many constraints. Post & Vilar [2018] propose dynamic beam allocation (DBA), a process that allows for an arbitrary number of constraints to be added while the added time complexity stays constant. They show, however, that DBA requires a rather high beam size of 10 in order to perform well, and argue that one of the problems is to find the correct permutations of the constraints, especially with many constraints, as the number of permutations increases exponentially. This problem is further aggravated by using the standard left-to-right decoding approach, as at the time of generating a token, the model has no knowledge about constraints that will have to be fulfilled at a later step in the translation process.

2.3 Unconstrained Terminology Enforcement

A radically different but successful approach by Dinu et al. [2019] does not concentrate on the decoding algorithm to place desired terms into the translation. Instead, the model learns to translate a word as the desired term, if the term is appended to the word in the source sentence. Compared to constrained beam search algorithms [Anderson et al., 2017; Hokamp & Liu, 2017; Post & Vilar, 2018], this method cannot be used after the training of an NMT system, as it needs to learn the mechanism during training. Conversely, after the model is trained, the so-called train-by-append method profits from practically no added decoding time, as the only added computation time is C added tokens in the source sentence, as well as an additional input-stream which indicates whether a (sub)word is a normal subword, part of a term in the source language or part of a term in the target language [Dinu et al., 2019]. This is especially impressive considering that their system reached term use rates of over 90% even though it used the transformer architecture in a comparatively small configuration with 2 encoder and decoder layers respectively. Additionally, in certain cases the model has learned to correctly inflect a term, even though only its base form was displayed to the model in the source sentence.

Although this approach promises to at least partly solve the inflection problem, for use in this thesis this approach is not suitable. The reason for this is because it is not possible to achieve a guarantee that the constraint will appear in the translation, as it is less of a constraint and more of a strong recommendation.

2.4 Sequence Generation Order

Since the advent of NMT, decoding has been mainly done with left-to-right generation. The standard transformer architecture [Vaswani et al., 2017] still produces state-of-the-art results using this technique. Although this method is intuitive and rather easily implemented in NMT, it is not ideal: the start of a generated translation cannot be modified later on in the sentence, and hence inherently constrains the generation of the rest of the translation to be of a certain grammatical structure. This is especially problematic during constrained decoding: if a constrained term needs to be inserted towards the end of the translation, the possibility is real that the term does not fit into the structure of the sentence. This can lead to sentences where the term appears twice in the translation, but in two different surface forms. Consequently, new decoding strategies have been introduced.

Gu et al. [2019] propose a Levenshtein Transformer, which does not generate the translation in a left-to-right fashion, but rather generates multiple tokens at the same time, and then subsequently improves the translation by inserting new tokens and deleting existing tokens. This process does not suffer from the above-mentioned problems of the base-Transformer using left-to-right generation.

Susanto et al. [2020] have demonstrated that the Levenshtein-Transformer [Gu et al., 2019] is well suited for lexically constrained decoding: Instead of starting with an empty sequence, the translation is instantiated with the desired terms. With this approach, the model refines the sequence by adding words around the terms and as a results of the terms being known from the beginning, the model can take them into account during all of the generation process. They report a term use rate of roughly 93.5% over different testsets using this method. To further increase the term use rate, they did not allow the deletion of said constraints, which occasionally happened. With this improvement, the term use rate amounts to 99%, and combined with not allowing tokens to be inserted amidst terms that span over more than one (sub)word, they reach a 100% certainty that constrained terms are included in the translation.

Additionally, it seems that disallowing the deletion of the constraints and insertion in-between them also comes with an improvement in translation quality, as these runs show the highest BLEU scores. This is, however, no surprise and to be taken with a grain of salt: their evaluation is based on an oracle experiment as it was proposed by Dinu et al. [2019]. In other words, the terms that are used as the lexical constraints are known to be part of the reference translation. As BLEU [Papineni et al., 2002] is an n-gram based metric, the scores are naturally expected to increase when a term is translated exactly as it appears in the reference. It also reduces the probability that the translation uses a different sentence structure than the reference translation, which further increases BLEU Scores without necessarily improving translation quality overall. Nevertheless, compared to DBA [Post & Vilar, 2018], this approach is a significant improvement when compared with other methods that provide high term use rates. Namely, constrained decoding with DBA can reach a term use rate very close to 100%, but in some cases only with a major decrease in inference time (Dinu et al. [2019] report a needed beam size of 20 to achieve a term use rate of 99%) and an accompanying loss in translation quality measured in BLEU as seen in Dinu et al. [2019]. Hence, moving away from the standard left-to-right decoding approach appears to benefit the cause of lexically constrained decoding. And, for professional translators in a post-editing setting, a guarantee that terminology guidelines are respected is certainly desirable.

Susanto et al. [2020] discuss two issues that remain when using lexical constraints with the Levenshtein Transformer: on the one hand, they did not propose a way to morphologically vary the constraints, which can provoke either artificially constructed sentenses to match the inflection of the term, or the inflection of the term not matching the rest of the sentence. On the other hand, when there is more than one term that should appear in the translation, their proposed Levenshtein transformer does not implement a way that allows the order of the constraints to be changed. As a result, a good translation depends on a good initialisation of the constraint order. However, as the number of permutations increases very fast ($n!$), a the probability of a good initialisation drastically decreases with the number of constraints. In the next section, I will address another variation of the Transformer architecture, which might have the capabilities to address these two issues.

2.5 Insertion Transformer with Unconstrained Generation Order

Emelianenko et al. [2019] propose an Insertion Transformer for unconstrained order sequence modeling (INTRUS). Whereas the Levenshtein Transformer [Susanto et al., 2020] operates in a non-autoregressive fashion, INTRUS decodes autoregressively, as it is the standard in NMT, which means that the translated sequence is generated piece by piece. Yet, in contrast to most state-of-the-art NMT systems, INTRUS can insert any token at any timestep at any desired position. This does come at the price of added computational complexity: The training duration of INTRUS is 3-4 times larger than that of a same-size standard Transformer [Vaswani et al., 2017]. At inference time, the speed of INTRUS is not harmed as much, as the inference speed is $O(|Y|^3k)$ compared to $O(|Y|^2k)$ for standard left-to-right decoding, which results in roughly 50% lower decoding speed on average, as $|Y|$ is rather small for most sentences [Emelianenko et al., 2019]. Nevertheless, 33% less performance is not negligible, and as the decoding time can easily explode for large sentences, this is certainly a disadvantage of INTRUS when compared to the work of Susanto et al. [2020]; Vaswani et al. [2017]. The reason for this exploding decoding time is that the decoder self-attention needs to be recomputed at every timestep because the insertion of new tokens changes the positional encodings of many tokens [Emelianenko et al., 2019].

Traditional left-to-right decoders predict the probability of the next word given the source sentence and the current unfinished translation. In comparison, INTRUS predicts the probability of the next insertion, which means it must predict the joint probability of a token and a position. More precisely, the probability distribution is computed as follows, as described in Emelianenko et al. [2019]:

$$\begin{aligned} p(\tau_t) &= p(token|pos) \cdot p(pos) \\ p(pos) &= softmax(H \times \omega_{loc}) \\ p(token|pos) &= softmax(h_{pos} \times W_{tok}) \end{aligned} \tag{2.1}$$

Compared to traditional left-to-right decoding, the probability of every next insertion τ_t is not simply dependent on the *token*, but also on the insertion position (*pos*). Thus, at each timestep t, there are t possible insertion positions. H is a matrix that contains t decoder hidden states h_{pos}, with each h_{pos} corresponding to one of the t possible insertion positions. ω_{loc} is a vector of trained weights that aims to predict the probability of each possible next insertion position. The softmax of the product of H and ω_{loc} hence yields a t dimensional vector containing probabilities for each pos. The most computationally intensive part is found in the last row: The probability of each token is not computed once, as it is done in traditional scenarios, but it is computed t times, once for each possible pos. This is achieved by taking each respective h_{pos} and multiplying it with a learned weight matrix W_{tok}, which predicts the probability of each token at position pos [Emelianenko et al., 2019].

Having the probability of every possible insertion makes INTRUS very attractive to use for lexically constrained decoding. In theory, the problem of inserting multiple constraints in the correct order can be left to INTRUS. After inserting the first constraint, the model computes the probabilities for the next constraint on both sides, and the constraint is inserted at the most probable location. Naturally, this also scales to more constraints as at each step, the model makes a decision as to whether the next constraint should be placed between two others or to one of the sides. This is a big advantage that INTRUS holds over the Levenshtein Transformer [Susanto et al., 2020].

INTRUS can also be used to judge which inflection of a constraint is the most probable. In order to make use of this ability, however, INTRUS needs to be provided with different inflections of the constraint. But, if provided with morphological variations of a constraint, the procedure to find out which variation is to be used is straightforward: INTRUS assigns a probability to each form of the constraint and the variation with the highest probability is chosen. Additionally, this should not interfere with the above-mentioned choosing of the correct order of constraints, i.e. it allows to have multiple constraints, and to have differently inflected forms for each of those constraints.

INTRUS shows great potential to achieve a guarantee that a term is in the source sentence while still being able to account for morphological variations of said term. Thus, I will use INTRUS and design a decoding algorithm that aims to leverage the advantages of an unconstrained sequence order generation as opposed to the traditional left-to-right decoding paradigm.

3 Materials

3.1 Parallel Data

The parallel data used for the training of the translation model was the publicly available data from the WMT18 English-German (En-De) news translation task, excluding the ParaCrawl corpus, as my available hardware resources were limited, and the addition of ParaCrawl significantly increases the data compared to the previous WMT17 news translation tasks. By excluding ParaCrawl, I achieve to have the same training data as it was used in the work of Dinu et al. [2019], which allows for a fair comparison. Once downloaded, the raw parallel sentences were normalized and tokenized using the Moses Toolkit [Koehn et al., 2007]. Afterwards, I used BPE [Sennrich et al., 2016] with 32000 merge operations to learn a joint vocabulary on the training data. Consequently, I applied the learned vocabulary to split the corpus into subword units. Finally I performed ratio-cleaning on the remaining sentences, where sentence pairs with a source/target ratio of 1.5 were removed, as were sentence pairs where one side exceeded the length of 250 subwords. This led to a training set with 2'115'613 sentence pairs, of which 500 were cut off to be used as validation set during training.

3.2 Testsets

To successfully evaluate the ability of INTRUS to choose the right morphological variant of a constraint, and how well it handles a noisy, I constructed a testset on the basis of the newstest2017 testset similarly to how Dinu et al. [2019] approached the evaluation of their system. However, as the oracle experiment artificially increases BLEU-Scores on the testset, I adapted my approach to generate a fairer testset. Analogously to Dinu et al. [2019], I download the IATE[1] termbase. I then extracted all the terms from all selectable domains into a 1:many (En:De) format, such that for a match on the En side of the translation, there would be many possible terms

[1]Available at `https://iate.europa.net`

to constrain. Additionally, I followed Dinu et al. [2019] and filtered out the entries where the English side was among the top 500 most frequent English words. To then annotate the testset, for every exact match on the En side, I lemmatized all corresponding De terms and the whole target sentence with spaCy [Honnibal & Montani, 2017], and then used the German lemma as constraint. With this measure, although it is still a form of an oracle experiment, I prevent that e.g. in certain cases the plural form is already used as a constraint. Using this method, I find 781 sentence pairs with a total of 981 valid constraints. This set is referred to as *oracle* in the Results section.

To test whether morphological variations improve the result, I generated different morphological variants as described in Section 3.3, and added all generated variants as allowed alternatives of the original constraint. Hence, the total number of constraints amounted to 4139. For terms where the morphology generation failed, the lemma was left as single constraint. This set is referred to *oracle+morphology* in the Results section.

To test whether INTRUS can pick the correct term when provided from among other, similar ones, I altered the allowed alternatives to be a randomly chosen De term that was in the corresponding En:De pair in the terminology database. In 949 cases, such an alternative existed, making it a total of 1930 valid constraints. In the Results section, this testset will be referred to as *oracle+noise.*

Finally, I combined both variants, i.e. I added morphological variation of all constraints, including the alternatives, which resulted in 7788 allowed constraints, which compete for the 981 positions in the target sentence. This set is named *morphology+noise* in the Result section.

3.3 Generation of Different Morphological Variants

In order to generate different morphological variants I used the precompiled finite-state transducer `zmorge-20150315-smor_newlemma.a`[2] from Zmorge [Sennrich & Kunz, 2014]. Zmorge is an open-source German morphological lexicon that is compatible with the finite-state grammar SMOR [Schmid et al., 2004]. Hence, I could use the precompiled transducer to generate various inflection forms of the constraints. Always starting from the lemma that was again found with spaCy [Honnibal & Montani, 2017], I then looped over all possible attributes of a certain word category:

[2]Available at `https://pub.cl.uzh.ch/users/sennrich/zmorge/`

- For nouns, all combinations of different cases and singular/plural variations.
- For verbs, I looped over all possible combinations of person, singular/plural, tenses, participles and modus.
- Finally, for adjectives, I looped over the possible combinations of adjective strength/weakness, comparation form, gender, singular/plural and cases.

Although occasionally an illegal form was generated, this should not cause a dramatic performance degradation, as at least most possible correct forms are generated, of which the model still has plenty to choose from.

4 Methods

4.1 Training

I trained an NMT system based on INTRUS for the language pair EN→DE. The model hyperparameters are as follows:

- Base learning rate: 1.4e-3
- Warmup steps: 16'000
- Maximum batch size: 4'000 tokens
- 8 Attention Heads
- 6 layers
- Feed-forward hidden layer size: 2048

These settings are the same settings that are described in Emelianenko et al. [2019]. The model is an encoder-decoder framework where the encoder is the same as proposed by Vaswani et al. [2017], and the decoder is the adapted version for unconstrained sequence order generation [Emelianenko et al., 2019].

The training procedure proposed by Emelianenko et al. [2019] includes two phases: a pretraining step and and a fine-tuning step. In the pretraining step, the model learns on uniformly distributed insertions, in order to not give preference to a certain order of generations. In the fine-tuning step, the model is consequently trained on samples from its own distribution, which has the goal of allocating the most probability mass along certain more preferable generation trajectories [Emelianenko et al., 2019]. While the fine-tuned model manages to outperform the baseline, the pretrained model performs 1.2 BLEU worse compared to the baseline and 2.9 BLEU worse compared to the fine-tuned model. Although state-of-the-art performance would be desirable, for the remainder of this thesis I only use a pretrained version of INTRUS, as fine-tuning could lead to performance degradation when the optimized trajectories do not have the constrained terms at the start position. Particularly, Emelianenko et al. [2019] note that the fine-tuned model has a high tendency to start the generation sequence with punctuation tokens or conjunctions, which is a very undesirable property for this thesis, as punctuation and conjunctions are hardly ever used as constraints.

The final translation model used for the evaluation was trained for 10^5 steps on a single RTX 2080Ti GPU. The total training duration was around 2 weeks.

4.2 Constrained Decoding with INTRUS

Source: Withdraw money at every ATM .
Reference: Hebe Geld an jedem Banc@@ omaten ab .
Constraints: (Geld, Geldes), (Banc@@ omat, Banc@@ omaten)

Step 1
Allowed insertions: **Geld**, **Geldes** or **Banc@@** at position 0

Step 2
Hypothesis: $_0$ Geld $_1$
Allowed insertions: **Banc@@** at position 0 or 1

Hypothesis: $_0$ Banc@@ $_1$
Allowed insertions: **omat** at position 1, **omaten** at position 1

Step 3
Hypothesis: $_0$ Geld $_1$ Banc@@ $_2$
Allowed insertions: **omat** at position 2, **omaten** at position 2

Hypothesis: $_0$ Banc@@ $_1$ omat $_2$
Allowed insertions: **Geld** at positions 0, 2, **Geldes** at positions 0, 2

Step 4:
Hypothesis: $_0$ Geld $_1$ Banc@@ $_2$ omaten $_3$
Allowed insertions: Every word from vocabulary at positions 0, 1, 3

Figure 1: Allowed example insertions at different timesteps

To perform constrained decoding with INTRUS I propose an algorithm (Algorithm 1) where during the first steps of decoding, the model is not allowed to generate subwords that are not included in the terms. I call this the constrained decoding phase, which lasts until all hypotheses in the beam meet all constraints.

Before taking a look at the algorithm, consider the example insertions in Figure 1: In this example, there are two constraint groups, namely (*Geld, Geldes*) and (*Banc@@ omat, Banc@@ omaten*). Before the model is allowed to generate any other tokens, the hypothesis needs to meet one constraint from each constraint group. Multi-subword constraints such as *Banc@@ omat* are always inserted in left-to-right order, which is why *omat* and *omaten* are not allowed insertions at timestep 1. Further, whenever a multi-subword constraint is started, the model is required to finish said constraint, before staring insertions of different constraints. This can be seen in the second example at timestep 2, where the only possible insertions are ones that finish the started constraint. Once a multi-subword constraint is finished, insertions in between its subwords is forbidden for the rest of the generation process, as can be seen in the last two examples. This ensures that a constraint appears in a demanded form in the translation.

Algorithm 1 Translation Step during Constrained Decoding phase with INTRUS (simplified)
Inputs: current beam *beam*, beam size *k*.
Output: new beam with best scoring hypotheses at next timestep

```
candidates ← [ ]
for hypo in beam do
  if hypo.is_incomplete() then
    if hypo.needs_continuation() then
      for possible constraint_continuation in hypo.continuations do
        candidates.append(hypo.insert_specific(constraint_continuation)
      end for
    else
      for possible constraint_start in hypo.starts do
        candidates.append(hypo.insert_specific(constraint_start)
      end for
    end if
  else
    candidates.append(model.generate(hypo, $k$)
  end if
end for
beam ←ARGMAX$_K$($k$, candidates.scores)
return beam
```

The simplified algorithm (Algorithm 1) makes use of some functions that I will now elaborate:

- **hypo.is_incomplete()** is an operation that checks whether this specific hypothesis already fulfills all the necessary constraints. Hence it returns **true** if the hypothesis does not fulfill the constraints and **false** otherwise.

- **hypo.needs_continuation()** checks whether the hypothesis contains incomplete parts of a constraint. It returns **true** if it contains an unfinished constraint and **false** otherwise. An unfinished constraint is a constraint that consists of multiple subwords, and its final subword has not yet been added to the hypothesis.
- **hypo.insert_specific()** is the way all constraints are inserted: It inserts a specific subword into the hypothesis at a specific position. From the output of the probability distribution, the probability of this particular insertion is found and the function returns a copy of the old hypothesis with updated sequence and probability, but leaving the original hypothesis unchanged. This is especially of essence when adding morphological variations, as it will be discussed in Section 4.3.
- **hypo.continuations** contains all currently possible continuations. Multiple possible continuations only exist when decoding with constraint groups (see section 4.3). A possible continuation is a tuple containing the continuation subword and its respective insertion position. Because constraints that consist of multiple (sub)words are inserted in a left-to-right manner, a continuation only has one possible insertion position, and that is directly to the right of the currently unfinished constraint.
- **hypo.starts** contains all possible starts for new constraints. If a constraint (group) in a hypothesis is satisfied, it no longer appears the hypo.starts of this hypothesis. New possible starts are allowed at all positions where an insertion does not split a previously placed constraint, and contain only one-subword constraints or the starting subword of multi-subword constraints. This porperty copies the No-Ins.-behaviour as seen in Susanto et al. [2020]
- **model.generate()** returns the k most probable insert operations that do not split previously placed constraints[1].

After the constrained decoding phase, i.e. as soon as all hypotheses contain all needed constraints, the decoding moves to the restricted open generation phase. In this phase a translation step consists only of a slightly modified **model.generate()**: Instead of computing the top k insertions for each individual hypothesis, the top k insertions considering the whole previous beam are generated.

[1]For efficiency reasons, I extract the top k + some buffer b possible insertions from the complete probability distributions (including insertions at disallowed positions), and then replace all insertions at disallowed positions with the next best allowed insertion from the buffer.

4.3 Adding Morphological Variation

Dinu et al. [2019] have proposed a method where the NMT model partly learns to correctly inflect a (semi-constrained) term. With constrained decoding, it is not entirely possible to leave the whole generation to the model, but as the model assigns probabilities to each constraint, it can judge which inflection of a term is the most probable, if it is provided with said inflection. Using Zmorge [Sennrich & Kunz, 2014], it is possible to generate inflected forms of German words. Thus, for each constraint, I generate its inflected terms as described in Section 3.2 and add them together with the original constraint to a single term group. This only minimally influences Algorithm 1: The only modifications needed are that **hypo.continuations** and **hypo.starts** respectively need to keep track of more possibilities. Consequently, **hypo.continuations** takes into account that a started constraint may have multiple continuations instead of one (i.e. when two inflections start with the same subword but end in different ones). And **hypo.starts** needs to take into account, that if a constraint and an inflected version of it start with different subwords, both insertion operations should be added to the possible candidates.

4.4 Simulating a Noisy Terminology Database

In the case where I simulate a noisy terminology database, no further adaptions need to be made, as a noisy term can simply be added to a term group in the same manner as a constraint and an inflected version of it. For the model, the mechanism of choosing from different subwords of the same word stem is exactly the same as choosing from less related subwords.

5 Results and Analysis

5.1 BLEU Scores

I report detokenized[1] BLEU scores that were computed with with the Moses Toolkit [Koehn et al., 2007]. Table 1 shows the BLEU Scores on the testsets described in Section 3. Table 2 shows the results on the testset described by Dinu et al. [2019]. It is the training set with 581 lines, where also approximate matches are used, which makes it necessary to inflect certain terms.

Experiment	BLEU
baseline	19.2
oracle	19.7
oracle+morphology	19.2
oracle+noise	19.5
oracle+noise+morphology	19.1

Table 1: BLEU scores using INTRUS with beam size 10

	BLEU IATE	BLEU no constraints
Dinu et al. (2019)	25.8	25.0
INTRUS (k=10)	20.8	18.1
INTRUS+morph (k=10)	20.6	18.1

Table 2: BLEU scores on IATE [Dinu et al., 2019]

5.2 Comparison to Previous Work

Although it was to be expected that the model doeall, the model performance much below the state of the art. A part of this comparably poor translation can be

[1]`multi-bleu-detok.perl`

attributed to the restricted computational resources for the training of the model. Additionally, as has already been shown in the work of Emelianenko et al. [2019], INTRUS does not reach state of the art performance when only using the pretraining procedure, although the loss in BLEU score in their work was 3 BLEU and not 7, as is the case for my model. When compared to the outputs of similar systems [Dinu et al., 2019; Susanto et al., 2020], the modifications made to improve constrained decoding hardly pay off. The upside is that when compared to previous beam search algorithm modifications such as DBA [Post & Vilar, 2018], the model performance measured in BLEU does not decrease when the terminology is constrained, as it could be observed in Dinu et al. [2019]. However, as Susanto et al. [2020] have reached BLEU scores of over 31 on the IATE testset with exact matches while also maintaining a 100% term usage rate, it is difficult to find an argument that speaks for using INTRUS with constrained decoding.

Although the testsets I created as described in Section 3.2 should require the use of morphological variation because the oracle terminology constraints are based on matching lemmas instead of exact matches, the BLEU scores are higher when when only using the lemma as constraint. A further analysis of this phenomenon is provided in Section 5.4.

On a positive note, it seems that INTRUS performs well in the scenario where a noisy alternative to the real constraint is added: *oracle+noise* and *oracle+noise+morphology* show only minor decreases in BLEU when compared to their non-noise counterparts, which is a remarkable result. Their respective non-noise counterparts likely present a form of BLEU upper-bound, because the added noise is very unlikely to improve BLEU scores, as the noisy terms specifically are not the desired translation of the constraint in this evaluation setting. And for this reason, it is quite a good result that BLEU only decreases by 0.2 and 0.1 respectively.

5.3 Examples & Issues

Table 3 shows an example where the lemmatisation with spaCy [Honnibal & Montani, 2017] fails, an issue that can be observed on quite a few occasions in the testset. In this instance, the adverb **klar** was incorrectly lemmatised to **klaren**, which is not a viable German word. The finite-state transducer then assumed it to be a verb and generated multiple forms, including **klare** but not **klar** (which would have been a lucky guess at a fictional imperative). However, in the case without added morphology, the resolution of this wrongly constrained term is remarkable, as the translation remains very accurate and grammatically correct. In the example with added morphology, **Aussagen** is also inflected correctly to match **klare**, but the beginning of the sentence does not fit.

		constraint
ref	Das hat der Hollywood-Star in einem Interview jetzt unmissverständlich **klar** gemacht.	
src	That is what the Hollywood star has made abundantly **clear** in an interview.	
oracle	Das hat der Hollywood-Filmstar in einem **klaren** Gespräch deutlich gemacht.	klaren
+morph	Das hat der Hollywood-Filmstar in einem Interview **klare** Aussagen gemacht.	klaren,klare,...

Table 3: Example of wrong lemmatisation

This example shows INTRUS's ability to model around the constraints that it has to work with, which is one of its advantages when compared to constrained beam search algorithms, where the part of the sentence before the constraint is modeled without accounting for it.

5.4 Performance Loss with Morphological Variations

Surprisingly, the BLEU scores show a slight decrease when morphological variants are added to the list of possible constraints. Because I specifically tried to avoid a scenario where choosing the base form would yield higher BLEU scores by matching terms using their lemma, the model should yield higher scores on *oracle+morphology* than on *oracle*, as it has the possibility to generate a correctly inflected constraint, whereas on *oracle* it is stuck with the base form of every term.

By taking a closer look at the testset, I found that the lemmatisation of spaCy [Honnibal & Montani, 2017] does not always manage to find the correct lemma, and in many cases in the testset, the supposed lemma that spaCy found was already a correctly inflected term according to the reference translation. To be precise, only in 182 segments out of the total 781 included in my testset contained at least one term where the base form in *oracle* did not exactly match the one in the reference translation. When taking this into account, the results reported in Table 1 are not very surprising, as for the majority of the test samples *oracle+morphology* is mostly adding noise to the constraints when compared to *oracle*. To make matters worse, when a different inflection is chosen than the one in the reference, this is likely to cause a change in the sentence structure and thus lead to loss in BLEU scores, disregarding whether the translation is actually correct. In Table 4, I report the BLEU scores on the 182 segments where the reference translation contains an inflected form of at least one constraint. On this subset, adding morphological variants of the constraints improves the BLEU score when compared to only using base form constraints.

Experiment	BLEU
oracle	16.2
oracle+morphology	16.9

Table 4: BLEU scores on the subset of the testset where constraints appear in an inflected form in the reference translation.

Hence, in situations where an inflection of a term is needed, my contribution does yield improvements in translation quality. However, in cases where only the base form of the constraint would be needed, adding morphological information results in lower BLEU scores, as the results from Table 1 show.

6 Conclusion

In conclusion, I contributed an algorithm that allows to perform lexically constrained decoding using an Insertion Transformer. Additionally, I introduced a mechanism that allows for a constraint to be a group of multiple valid options, of which only one needs to be in the final translation. This allows let the model decide which inflection form shall be used when a constraint group consists of multiple possible inflected forms of the same term, or to let the model decide which term to use when there is a conflicting entry in the term database.

My first research question aimed to compare the use of constrained decoding with the Insertion Transformer [Emelianenko et al., 2019] to other constrained decoding methods used in NMT. In order to find an answer I proposed and implemented Algorithm 1. My experiments have shown that overall, my model performs significantly worse than the previous methods proposed by Dinu et al. [2019] and Susanto et al. [2020] and fails to produce state-of-the-art results. Although this is partly caused by the restricted resources I had for the training of my model, this alone does not explain such a big loss in translation quality. Still, using lexical constraints with the Insertion Transformer does come with some benefits: first, compared to Dinu et al. [2019], my method achieves a guarantee that the constraint appears in the translation and does not need a terminology database to be included in the training process. Secondly, in contrast to Susanto et al. [2020], it allows the placement of constraints in any possible order.

The second research question of this thesis was concerned with whether morphological variations of lexical constraints improve translation quality. In a first experiment, the results pointed towards a negative answer, as on the initial testset the BLEU scores slightly decreased. However, upon an investigation of the testset, I was able to to show that in cases where an inflected lexical constraint is desired, morphological variations of the constraints increase BLEU by 0.7.

To answer the last research question, namely whether using an Insertion Transformer is beneficial in cases where a terminology database contains noise such as duplicate terms, I conducted experiments where the model was allowed to choose between the reference translation of the constraint and a noisy duplicate entry from the term database. The results revealed that the translation quality is only suffering minor decreases with the added noise. This is, to the best of my knowledge, the first work that provides a method where the model can work with a noisy translation database and decide at inference time which of the provided constraints should be chosen.

Further research could incorporate the train-by-append approach from Dinu et al. [2019] to the INTRUS training procedure. This would eliminate a theoretical disadvantage that this work has in comparison: Although the model is provided with the constraints from the beginning of the decoding process, it has no explicit information to which term in the source sentence the constraint corresponds. The addition of such explicit information might provide an advantage in choosing the correct inflection, as the model then has some information where a constraint is positioned in the source sentence, which can be a good indicator for its position in the translation, and thus also its inflection.

References

Anderson, Peter, Basura Fernando, Mark Johnson & Stephen Gould. 2017. Guided Open Vocabulary Image Captioning with Constrained Beam Search. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 936–945. Copenhagen, Denmark: Association for Computational Linguistics. doi:10.18653/v1/D17-1098. `https://www.aclweb.org/anthology/D17-1098`.

Dinu, Georgiana, Prashant Mathur, Marcello Federico & Yaser Al-Onaizan. 2019. Training Neural Machine Translation to Apply Terminology Constraints. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 3063–3068. Florence, Italy: Association for Computational Linguistics. doi:10.18653/v1/P19-1294. `https://www.aclweb.org/anthology/P19-1294`.

Emelianenko, Dmitrii, Elena Voita & Pavel Serdyukov. 2019. Sequence Modeling with Unconstrained Generation Order.

Gu, Jiatao, Changhan Wang & Jake Zhao. 2019. Levenshtein Transformer.

Hasler, Eva, Adrià de Gispert, Gonzalo Iglesias & Bill Byrne. 2018. Neural Machine Translation Decoding with Terminology Constraints. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, 506–512. New Orleans, Louisiana: Association for Computational Linguistics. doi:10.18653/v1/N18-2081. `https://www.aclweb.org/anthology/N18-2081`.

Hokamp, Chris & Qun Liu. 2017. Lexically Constrained Decoding for Sequence Generation Using Grid Beam Search. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1535–1546. Vancouver, Canada: Association for Computational Linguistics. doi:10.18653/v1/P17-1141. `https://www.aclweb.org/anthology/P17-1141`.

Honnibal, Matthew & Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Koehn, Philipp, 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 & Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, 177–180. Prague, Czech Republic: Association for Computational Linguistics. `https://www.aclweb.org/anthology/P07-2045`.

Koehn, Philipp, Franz J Och & Daniel Marcu. 2003. Statistical phrase-based translation. Tech. Rep. UNIVERSITY OF SOUTHERN CALIFORNIA MARINA DEL REY INFORMATION SCIENCES INST.

Papineni, Kishore, Salim Roukos, Todd Ward & 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*, 311–318. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics. doi:10.3115/1073083.1073135. `https://www.aclweb.org/anthology/P02-1040`.

Post, Matt & David Vilar. 2018. Fast Lexically Constrained Decoding with Dynamic Beam Allocation for Neural Machine Translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 1314–1324. New Orleans, Louisiana: Association for Computational Linguistics. doi:10.18653/v1/N18-1119. `https://www.aclweb.org/anthology/N18-1119`.

Schmid, Helmut, Arne Fitschen & Ulrich Heid. 2004. SMOR: A German Computational Morphology Covering Derivation, Composition and Inflection. In *LREC*, 1–263. Lisbon.

Sennrich, Rico, Barry Haddow & Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1715–1725. Berlin, Germany: Association for Computational Linguistics. doi:10.18653/v1/P16-1162. `https://www.aclweb.org/anthology/P16-1162`.

Sennrich, Rico & Beat Kunz. 2014. Zmorge: A German Morphological Lexicon Extracted from Wiktionary. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, 1063–1067. Reykjavik, Iceland: European Language Resources Association (ELRA). `http://www.lrec-conf.org/proceedings/lrec2014/pdf/116_Paper.pdf`.

Susanto, Raymond Hendy, Shamil Chollampatt & Liling Tan. 2020. Lexically Constrained Neural Machine Translation with Levenshtein Transformer.

Sutskever, Ilya, Oriol Vinyals & Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence & K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 27*, 3104–3112. Curran Associates, Inc. `http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf`.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser & Illia Polosukhin. 2017. Attention is All You Need, `https://arxiv.org/pdf/1706.03762.pdf`.