Mitra: Improving Terminologically Constrained Translation Quality with Backtranslations and Flag Diacritics

Iikka Hauhio[†‡] and **Théo Friberg**[†‡]
[†] Kielikone Oy, Helsinki, Finland
[‡] Department of Computer Science, University of Helsinki, Finland
{iikka.hauhio,theo.friberg}@kielikone.fi

Abstract

Terminologically constrained machine translation is a hot topic in the field of neural machine translation. One major way to categorize constrained translation methods is to divide them into "hard" constraints that are forced into the target language sentence using a special decoding algorithm, and "soft" constraints that are included in the input given to the model.

We present a constrained translation pipeline that combines soft and hard constraints while being completely model-agnostic, i.e. our method can be used with any NMT or LLM model. In the "soft" part, we substitute the source language terms in the input sentence for the backtranslations of their target language equivalents. This causes the source sentence to be more similar to the intended translation, thus making it easier to translate for the model. In the "hard" part, we use a novel nondeterministic finite state transducer-based (NDFST) constraint recognition algorithm utilizing flag diacritics to force the model to use the desired target language terms.

We test our model with both Finnish–English and English–Finnish real-world vocabularies. We find that our methods consistently improve the translation quality when compared to previous constrained decoding algorithms, while the improvement over unconstrained translations depends on the familiarity of the model over the subject vocabulary and the quality of the vocabulary.

1 Introduction

In this paper, we present *Mitra*, an end-to-end pipeline for terminology-constrained translation that combines a novel constrained beam search algorithm with backtranslation substitution.

Terminology-constrained machine translation is a popular topic in the field of machine translation, and has been a focus of several shared tasks in the WMT conference (Alam et al. 2021b; Semenov et al. 2023). In constrained translation, the system is given a lexicon, or a terminology, and it must use the words given in this terminology when translating sentences. While this was a trivial task in phrase-based statistical (cf. Koehn et al. 2003) and rule-based machine translation systems (cf. Arnola 1996), implementing it for neural systems has proved to be much more difficult due to their black-box nature.

The existing methods can be divided into the so called "hard" and "soft" constraints. Hard constraints use *constrained decoding* algorithms such as constrained beam-search (Hokamp and Liu 2017; Anderson et al. 2017), which first decides on the acceptable forms of constraints at the token level and then forces the decoder of an NMT system to abide by them. Soft methods, on the other hand, use a neural network specifically trained for the purpose of constrained translation, and the constraints can be given to the encoder of the network as input (cf. Bergmanis and Pinnis 2021). Both of these methods have their own advantages: hard constraints can be enforced on any neural network without the need to train or fine-tune anything, while soft constraints are generally faster. Typically, hard constraints only mandate that terms occur somewhere in the translated sentence, while soft constraint approaches allow an explicit coupling of the source and target terms.

Implementing hard constraints for agglutinative languages has several difficulties. Foremost, if a term has multiple possible inflected forms, and the correct form is not known beforehand, the constrained decoding algorithm must be given multiple alternative forms (Anderson et al. 2017), a feature not widely supported by many algorithms such as Post and Vilar (2018); Hu et al. (2019). Moreover, the system must be able to generate these alternative forms, requiring the use of language-specific morphological generators, which might not be available for all languages.

Another problem of hard constraints is that the translation quality might be very poor if the machine translation model does not recognize the constrained words, a situation which in our experience is very common as terminologies often contain uncommon technical jargon, brand names, and other terms not appearing in the training data. In our specific case, we tested our method with a vocabulary provided by the Finnish Forest Centre containing names of insects in Finnish and English (Metsäkeskus 2023). Many of the names have surprising translations: for example, a "violet tanbark beetle" is called "papintappaja" in Finnish, which means "priest-slayer" if translated literally. If the translation model has not seen this term before, it cannot correctly translate it without terminology constraints. Even then, the model has a hard time determining the correct location for the constraint in the output sentence (cf. Hasler et al. 2018). See Section 2.3 for more details of this problem.

We propose a combined method that tackles both of the aforementioned problems. To support heavily-inflected languages such as Finnish, we introduce a "hard" finite-state automaton-based constraint recognition algorithm that can recognize arbitrarily large disjunctive constraints. For the problem of expressions that were not encountered at training time, we propose a "soft" backtranslation substitution algorithm that makes it possible to use terminology constraints even when the neural network sees no connection between the source-language term and the target-language term. Both of these methods are integrated into an end-to-end pipeline that takes source-language sentences and lexicons as input and produces lexically constrained translations. We argue that the "hard" and "soft" constraint methods complement each other and work together as a whole greater than the sum of its parts.

In this paper, we first describe existing constrained beam search algorithms (Section 2). We then give an overview of our pipeline, including detailed descriptions of the backtranslation and the constraint recognition algorithms (Section 3). Finally, we evaluate these algorithms against the existing algorithms (Section 4).

2 Constrained Beam Search

"Hard" terminology constraints refer to phrases (i.e. sequences of tokens) that are forced to appear in an output sequence during beam search. While a regular beam search compiles the list of new hypotheses by finding the most probable continuations for the current hypotheses (Koehn 2009), a constrained beam search algorithm additionally proposes tokens in the constraints as possible continuations (Hokamp and Liu 2017). Several algorithms exist, differing mainly in beam allocation (i.e. how much of the beam is reserved for hypotheses containing constraints), and constraint recognition (i.e. how they determine which constraints are fulfilled and propose constraint tokens as continuations).

2.1 Existing Algorithms

Hokamp and Liu (2017) present an algorithm called Grid Beam Search (GBS), which allocates $C + 1$ hypothesis banks in the beam, where C is the number of constraint tokens. Each hypothesis in bank $i \in [0, C]$ must have exactly i fulfilled constraint tokens. Unlike all other algorithms inspected, they do not allow backtracking, i.e. constraints that were previously considered fulfilled can not become unfulfilled again: if a model begins generating a prefix of a multi-token constraint, the hypothesis can only be continued by generating the rest of the constraint. In other words, hypotheses can only move upwards in the banks or stay at the same level. This also means that GBS does not require a constraint recognition algorithm for detecting which constraints are currently fulfilled in a hypothesis – it only needs to remember which constraint tokens it has previously generated, since those tokens will stay fulfilled.

Post and Vilar (2018) criticize GBS for its beam allocation, as the number of hypotheses grows linearly with the number of constraints. They propose a method called Dynamic Beam Allocation (DBA), in which the beam size is constant, and the hypothe-

Original sentence	**Metsäpaloregiimi** summaa yhteen lähes kaikki metsäpaloihin vaikuttavat tekijät.
Greedy tracking	The forest fire[†] **forest fire regime** brings together almost all the factors that affect forest fires.
Exact tracking	The **forest fire regime** brings together almost all the factors affecting forest fires.

Table 1: Having reached the point marked with †, greedy tracking accepts the constraint `metsäpalo` →`forest fire` and discards the start of the constraint `metsäpaloregiimi` →`forest fire regime`. See also Appendix D.1.1.

Original sentence	[. . .] rajoittamaan ja supistamaan paloa **rajoituslinjojen** avulla.
Natural translation	[. . .] limit and reduce the fire by means of **firebreaks**.
Machine translation	[. . .] limit and reduce the **firebreak** by means limiting lines
Original sentence	Raivaamalla tehtyjä **rajoituslinjoja**$_1$ ovat **palokuja**$_2$ ja **palokäytävä**$_3$.
Natural translation	**Fire lines**$_2$ and **fire alleys**$_3$ are **firebreaks**$_1$ made by clearing.
Machine translation	The **firebreaks**$_1$ and **fire alley**$_3$ are **fire lines**$_2$ made by clearing.

Table 2: Examples of leaks and misplacements. In the first sentence, the model leaks the structure of "rajoituslinja" (lit. limiting line). In the second sentence, the model exchanges the constraints, changing the meaning of the sentence.

ses are assigned to different numbers of fulfilled constraints dynamically, making GPU memory optimization easier. In addition, to enable backtracking, they give a detailed description of a table-based data structure used for constraint recognition. The table contains information on which tokens are part of multi-token constraints and which of them are fulfilled. If the algorithm generates a token that is not a continuation of the current constraint being generated, it backtracks by marking the previously generated tokens unfulfilled.

This algorithm is further improved by Hu et al. (2019) who note that the constraint recognition algorithm proposed by Post and Vilar is flawed, as it cannot properly recognize overlapping constraints. They propose a trie-based algorithm claimed to resolve these issues. They also detail a method that allows sorting and selecting hypotheses completely in the GPU memory, further decreasing the overhead of the algorithm. Neither Post and Vilar (2018) nor Hu et al. (2019) support disjunctive constraints required by heavily-inflected languages, although we note that either of the algorithms can be relatively easily expanded to support them.

In addition to tables and tries, finite-state automata can be used to recognize fulfilled constraints in a hypothesis (Anderson et al. 2017; Hasler et al. 2018). This approach also supports disjunctive constraints and multi-token constraints.

2.2 The Problem of Greediness

The constraint recognition algorithms proposed by Post and Vilar (2018) and Hu et al. (2019) are greedy, which allows them to operate in $O(n)$ time. While this is good for time complexity, it also makes the algorithms incorrect: they cannot detect some valid sequences that contain overlapping constraints (see Table 1 for an example).

We assert that no greedy algorithm can detect all valid sequences. Consider the following sequences: $abcde{*}abcd$ and $abcde{*}cdeab$, with the constraints ab, cde, and $abcd$. $*$ represents a sequence of arbitrary tokens. Consider a greedy algorithm that has processed the first five tokens ($abcde$), as shown below:

	`abcde*`
Interpretation 1:	`ab`
	`  cde`
Interpretation 2:	`abcd`

Due to greediness, the algorithm must pick one of the two interpretations: the beginning of the string contains the constraints ab and cde, or it contains the constraint $abcd$. As the greedy algorithm does not backtrack, the end of the string cannot be taken into account. However, the correct interpretation depends on how the string ends. If the ending is $abcd$, the first interpretation was correct. On the other hand, if it is $cdeab$, the second interpretation was. Thus, all greedy algorithms fail to detect at least one of $abcde * abcd$ and $abcde * cdeab$.

The finite-state automaton-based approach suggested by Anderson et al. (2017) does not suffer from greediness, but is, as presented, infeasible in our use case: the number of hypotheses is 2^C, growing exponentially as the number of constraints C is increased.

2.3 Rare and Obscure Terms

Our method tackles what we have termed *obscure terms*. These are terms that are completely unpredictable to the model being decoded, as they are lexically surprising and have not been seen at training time. We argue that the challenge posed by obscure terms is fundamentally different from synonym selection in constrained decoding: not only are we willingly sampling along suboptimal paths, but we also sometimes have to choose tokens that directly contradict the analysis of the language model.

The nature-related lexica (Metsäkeskus 2022, 2023) we used during development and evaluation were full of these obscure terms. The Forest Centre terminologies contained such terms as “papintappaja” (violet tanbark beetle, lit. “priest-slayer”) and “tukkimiehentäi” (large pine weevil, lit. “log-man’s louse”). These target terms are very unexpected and clearly unfamiliar to the model, as seen from the low term accuracy of the unconstrained translation model in our evaluation (see Section 4).

Obscure terms produced characteristic failures. As the model evaluates all positions for a constraint to be unlikely, it will often result in *misplacements* as well as *leaks* where a part of the structure of a source language constraint still made its way to the translation. See Table 2 for an example of both.

3 The Mitra Pipeline

We designed our constrained translation pipeline with three goals: 1) allowing a high number of disjunctive constraint alternatives to support highly agglutinative languages such as Finnish, 2) fix the problems caused by greediness in the previous constraint recognition algorithms, and 3) make constrained decoding a viable alternative even for rare terms not present in the training data of the neural network. The goals 1 and 2 are fulfilled by using a custom finite-state automaton-based constraint recognition algorithm, while goal 3 is fulfilled by a backtranslation substitution algorithm.

The full end-to-end pipeline contains the following components:

1. **Term Recognition and Constraint Generation.** A dependency parser is used to extract the noun, verb, and adjectival phrases contained in the input sentence. If any of the phrases is found in the lexicon, all supported inflected forms of the target-language term are generated and added as a constraint.

2. **Backtranslation Substitution.** Each of the target-language terms added as constraints is translated back to the source language using an NMT model trained on the same data as the model used to perform the actual translation. The input sentence is modified so that the recognized terms are replaced with the backtranslations, inflected and capitalized similarly to the original terms.

3. **Constrained Beam Search.** A constrained beam search is performed to translate the input sentence to the target language.

We implement the pipeline for Finnish, Swedish, and English in all language directions. As our pipeline is agnostic to the NMT model itself, it can be used with any model as long as the appropriate language-specific modules have been implemented.

3.1 Term Recognition and Constraint Generation

Term recognition refers to scanning the input sentence and detecting all the phrases in the sentence that also appear in the terminology. We provide a phrase detection module for each of the supported languages. Each of the modules first performs the following high-level steps, although the specific methods are highly language-dependent and not within the scope of this paper.

1. **Dependency parsing.** We use the Stanford Stanza Python package (Qi et al. 2020) for English and Swedish, and the TranSmart dependency parser (Nykänen 1996) for Finnish.

2. **Phrase detection.** We iterate the dependency tree recursively and for each noun, adjective, and verb, we construct a list of noun phrases, adjectival phrases and verb phrases, respectively, as explained in the next step.

3. **Dependent selection.** For each noun, we iterate all combinations of its adjectival dependents (with the limit up to 6 adjectival dependents). We assemble a list of these combinations. For example, if the phrase is “young, strong cat”, we would generate the combinations “cat”, “young cat”, “strong cat”, and “young, strong cat”. For Finnish, we also return parts of compound nouns. For adjectives and verbs, we do not include any dependents, and simply return a one-item list with the adjective or verb itself.

		Recognized terms and backtranslations
Original sentence	**Hosat** ovat käteviä työkaluja.	{hosa → fire swatter}
Backtranslation	**Paloswatterit** ovat käteviä työkaluja.	{paloswatteri ← fire swatter}
Translation	**Fire swatters** are handy tools.	
Original sentence	The characteristics of the **live fuel type** are defined mainly on the basis of the tree stand and ground vegetation.	{live fuel type → palokasvustotyyppi}
Backtranslation	The characteristics of the **type of fire growth** are defined mainly on the basis of the tree stand and ground vegetation.	{type of fire growth ← palokasvustotyyppi}
Translation	**Palokasvustotyypin** ominaisuudet määritellään pääasiassa puuston ja maakasvillisuuden perusteella.	

Table 3: Example of phrase detection and backtranslation substitution.

4. **Lemmatization.** We lemmatize each phrase returned by the previous step. For Finnish and Swedish, this includes taking into account adjective-noun agreement: Finnish nouns agree in case and number, Swedish nouns in determinateness, gender and number.

After phrase detection, we compare the list of phrases to the terminology and generate a list of constraints. For each term, the disjunctive constraint has multiple alternatives corresponding to the different inflectional forms of the term. Similarly to above, we perform dependency parsing for the target-language terms, and then inflect them taking into account adjective-noun agreement. For Finnish, we do not generate all the possible forms due to their high number, instead we have a list of the most common forms.

We use several open-source[1] and proprietary[2] language modules. Details of this step are not within the scope of this paper.

3.2 Backtranslation Substitution

After phrase detection, we perform backtranslation for all target language terms that have been added as constraints by using a reverse-language NMT model trained with the same dataset as the model used for translation proper. In our experiments, we use the Opus-MT Tatoeba Challenge models (Tiedemann 2020) which include models for both language directions for most language pairs.

After producing the backtranslations, we verify that they improve the translation quality by using the NMT model to calculate scores for both (original term → target term) and (backtranslation → target term) pairs. If the pair with the backtranslation yields a better score, we replace the original term in the source language sentence with the backtranslation. We use dependency parsing and the language-specific modules detailed in the previous section to inflect the backtranslation in the same form as the original term. We also match the initial letter case.

An example of the backtranslation substitution process is given in Table 3.

3.3 Constrained Beam Search

We use a constrained beam search algorithm very similar to the one described by Post and Vilar (2018). The details of our algorithm are presented in Appendix A. The main differences to the previous algorithms are in our constraint recognition algorithm, detailed in the following section.

3.4 Constraint Recognition during Beam Search

For constraint recognition, we adapt a finite-state automata (FSA)-based approach similar to Anderson et al. (2017). We note that while the method used by Anderson et al. requires 2^C hypotheses, a different beam allocation strategy such as the one used in Grid Beam Search (Hokamp and Liu 2017) or Dynamic Beam Allocation (Post and Vilar 2018)

[1] We use the Python packages stanza, pyvoikko (for lemmatization of Finnish compound words), pyomorfi (for inflecting Finnish verbs), taivutin (for inflecting Finnish nominals), inflex (for inflecting English). For Swedish, we use a proprietary statistical guesser based on the Saldo inflecto (https://github.com/kielikone/saldo-inflector).

[2] Mostly TranSmart pipeline components (Nykänen 1996)

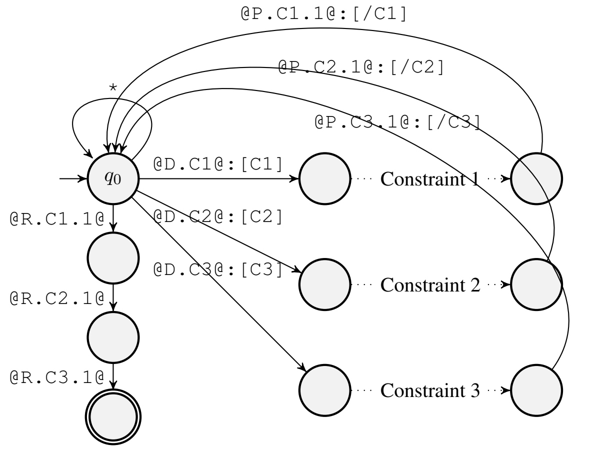

Figure 1: The structure of the finite state transducer that can recognize three constraints in any order. To reduce the size of the transducer, we use flag diacritics (Beesley and Karttunen 2003, chapter 8). @D.f@ is a flag diacritic that succeeds if f is undefined. @P.f.1@ is a flag diacritic that always succeeds and sets $f = 1$. @R.f.1@ is a flag diacritic that succeeds if $f = 1$. The nodes inside the constraints are not included; they would form a trie matching to all alternatives of that disjunctive constraint. The output of the transducer is the input, with symbols added for marking starts ([C1]) and ends ([/C1]) of constraints.

may also be used, which drops the beam size to either $O(C)$ or $O(1)$, respectively. In these scenarios, not every possible combination of constraints is stored in the beam: only the ones with the highest scores. We use a non-deterministic FSA that can track all possible interpretations at the same time: the two interpretations that would have been tracked in entirely different hypotheses in their solution can be tracked with a single hypothesis in our solution.

While the beam size can be limited to be linear with regard to the number of constraints, the memory constraints of the finite-state automaton cannot be. If the automaton is deterministic, its size is $O(2^C)$ in the worst-case scenario[3]. If the automaton is non-deterministic, its size will be $O(C)$, but the number of simultaneous states might be $O(2^{C \max |C_{i,j}|})$, i.e. in the worst case the number of states grows exponentially with regard to the number of constraints and their lengths. To mitigate this issue, we implement an optimization that removes most of the simultaneous states when we can safely determine that they recognize the same set of strings. This optimization is detailed below, after we have detailed the structure of the FSA.

Figure 1 has a graph of a finite-state transducer that recognizes three constraints in any order. Since the number of orderings the constraints can be in is $C!$, we employ flag-diacritics (Beesley and Karttunen 2003, chapter 8) to reduce the number of nodes in the transducer. Flag diacritics behave like epsilon edges, but can only be followed if a variable, called a *flag*, is set to a specific value. In our transducer, each constraint has its own flag. The parts of the transducer matching to constraints are fenced with the @D.f@ diacritics that succeed only if the flag f is undefined, thus preventing the transducer recognizing any constraints more than one time. After a constraint is fully recognized, the @P.f.1@ diacritic is used to set the flag value to 1. The accepting node of the transducer is fenced with diacritics of type @R.f.1@ that require that the flag must be set to value 1.

The specific type of finite-state automata we use is the finite-state transducer, in which each edge can both consume an input symbol and output a symbol (Beesley and Karttunen 2003). By using a finite-state transducer instead of a regular finite-state machine, we detect the beginnings and ends of the constraints. The transducer outputs each input symbol, and additionally provides a start token (such as [C1]) and an end token (such as [/C1]) for each constraint. When we discuss "states" below, we refer to (q, m) pairs, in which q is a node in the finite-state machine, and m is the sequence of output symbols produced.

Since the transducer is non-deterministic, the number of parallel states can grow exponentially. For example, if there are C constraints and all of them are present in the input string, in the end there will be at least 2^C simultaneous states: one in which none of the constraints matched, one in which all of them matched, and all the possible combinations in between. To prevent this, we remove some of the states between each iteration based on the following condition: if two states S_1 and S_2 are both in q_0 (the initial state as in Figure 1), and the set of fulfilled constraints in the output of S_1 is a proper subset of fulfilled constraints in S_2, the state S_1 is removed. The proof that this does not change the set of strings that are accepted by the FSA is included in Appendix B. This optimization makes the finite-state automaton computationally feasible on the real-world data we used.

The finite-state machines were implemented using the kfst Python package.

[3] Since all possible combinations of recognized constraints (2^C) must be represented.

Configuration Name	Decoding Algorithm	Constraint Recognition	Backtranslations
Mitra-FB	CBS	NDFST	Yes
Mitra-TB	CBS	Trie	Yes
Mitra-F	CBS	NDFST	No
Mitra-T	CBS	Trie	No
Mixtral	Sampling ($T = 0.2$)	N/A	No
Poro	Sampling ($T = 0.2$)	N/A	No
Unconstrained	Greedy	N/A	No

Table 4: Evaluated configurations of the pipeline. CBS refers to our Constrained Beam Search algorithm as described in this paper. NDFST is our non-deterministic finite-state transducer. Trie refers to the constraint recognition algorithm inspired by Post and Vilar (2018); Hu et al. (2019) modified to support disjunctive constraints. In addition to our pipeline, we use Mixtral (Jiang et al. 2024) and Poro (SiloAI 2023), both large language models, and unconstrained machine translations. All methods apart from Mixtral and Poro use Opus-MT Tatoeba Challenge models for Finnish and English (Tiedemann 2020).

4 Evaluation

We use both automatic evaluation and human evaluation to measure the quality of the translations produced by our pipeline. The automatic methods include BLEU, chrF, TER, and COMET scores, as well as measuring the number of fulfilled constraints. In human evaluation, we asked a professional translator to evaluate all translated sentences and mark them either as OK, erroneous due to incorrectly applied constraints, or erroneous due to other cause.

Due to time constraints, we did not evaluate the Swedish translation even though we implemented it.

4.1 Evaluated Pipeline Configurations

For evaluation, we prepared four configurations of the Mitra pipelines: half of them use the NDFST-based constraint recognizer, and half of them a trie-based recognizer inspired by (Post and Vilar 2018; Hu et al. 2019) modified to support disjunctive constraints. Both of the two algorithms are evaluated with and without backtranslation substitution, with parameters $B = 1, k = 5, S = M = \infty$ and a 120 s timeout. In addition, we translated the sentences without constraints or backtranslations. In each case, we use the Opus-MT Tatoeba Challenge models for Finnish and English[4] (Tiedemann 2020).

We also compare our methods to general-purpose language models Poro[5] (SiloAI 2023) and Mixtral[6] (Jiang et al. 2024) by embedding the constraint words into the prompt (cf. Ghazvininejad et al. 2023). For these models, we use the term recognition and constraint generation components of our pipeline, but do not apply backtranslations or enforce hard constraints. The prompts used are listed in Appendix C. We tried both zero-shot and 2-shot prompting and found that to achieve best results BLEU-wise, Mixtral needed to be prompted in a zero-shot manner and Poro in a 2-shot manner.

We ran the Opus-MT models on a Tesla T4 GPU, and the Poro and Mixtral models on an A100 80GB GPU.

We used greedy decoding with the unconstrained translation and temperature sampling with the LLMs, since these are the sampling methods most often used with these models. While beam search could have been used for both of these, and could have improved their performance, deciding the fair beam size would not have been trivial: the constrained translation has beam size $C + 1$, where C is the number of constraint tokens. If no constraints are used, this results in the beam size of 1, which corresponds to greedy decoding. Therefore, to simplify our experiment, we decided to not increase the beam size over this default unconstrained size of 1.

To save human evaluation resources, we performed the human evaluation for the model that received better BLEU scores, which is Mixtral for the Finnish–English translation direction, and Poro for the English–Finnish translation direction.

4.2 Evaluation Corpus

We conducted the evaluation on two vocabularies listed below. For both of them, we used the head words of the entries to construct the translation terminology and the definitions of the entries as the test sentences.

[4] https://huggingface.co/Helsinki-NLP/opus-mt-tc-big-fi-en and https://huggingface.co/Helsinki-NLP/opus-mt-tc-big-en-fi

[5] https://huggingface.co/LumiOpen/Poro-34B, 700B variant

[6] https://huggingface.co/mistralai/Mixtral-8x7B-Instruct-v0.1

	Forest Fires (EN–FI)					Finnish Parliament (FI–EN)				
Configuration	BLEU	chrF	TER	COMET	Acc.	BLEU	chrF	TER	COMET	Acc.
Mitra-FB	**20.96**	**62.31**	**69.62**	**0.90**	**100.00**	35.50	63.77	53.57	0.85	98.34
Mitra-TB	19.70	59.35	72.10	0.89	94.83	**35.70**	**64.33**	**53.48**	0.84	**100.00**
Mitra-F	19.48	61.87	70.09	0.89	**100.00**	35.21	63.67	54.45	0.85	98.34
Mitra-T	19.13	59.19	75.06	0.89	94.83	35.40	64.12	54.61	0.84	**100.00**
Mixtral	7.71	46.49	92.43	0.79	84.48	32.99	62.85	55.71	0.84	82.82
Poro	16.01	55.95	79.20	0.88	83.62	21.13	55.38	70.71	0.84	50.27
Unconstrained	16.45	54.55	76.95	0.86	22.41	34.48	62.08	53.52	**0.86**	49.46

Table 5: The results for automatic evaluation of the configurations described in Table 4. "Acc." refers to the number of fulfilled constraints.

	Forest Fires (EN–FI)			Finnish Parliament (FI–EN)		
Configuration	OK %	Constr. error %	Other error %	OK %	Constr. error %	Other error %
Mitra-FB	**60.00**	2.35	37.65	60.56	10.83	28.61
Mitra-TB	55.29	4.70	40.00	57.78	13.61	28.61
Mitra-F	55.29	9.41	**35.29**	56.39	15.00	28.61
Mitra-T	52.94	9.41	37.65	54.17	17.78	**28.06**
Mixtral				66.11	**0.56**	33.33
Poro	52.94	2.35	44.71			
Unconstrained	32.94	**0.00**	67.06	**67.78**	0.83	31.39

Table 6: The results for human evaluation of the configurations described in Table 4. All of the sentences were categorized to the three categories "OK", "Erronous due to incorrectly applied constraints" (Constr. err), and "Erronous due to other error". The timed out sentences are counted towards other errors.

1. **Forest Fire Vocabulary** by the Finnish Forest Centre (Metsäkeskus 2022), consisting of 85 Finnish/English word pairs and definitions. For this vocabulary, we translated the definitions in the English–Finnish direction.

2. **Finnish Parliament Vocabulary** by the Finnish Parliament (Eduskunta 2008), consisting of 360 Finnish/English word pairs and definitions. We used only 358 of these since two contained special characters for which the preprocessing pipeline failed. For this vocabulary, we translated the definitions in the opposite direction: Finnish–English.

We release all term pairs and test sentences openly[7].

4.3 Evaluation Methods

We performed both automatic and human evaluation. For automatic evaluation, we calculated BLEU, chrF, and TER scores for the sentences using the `sacrebleu` Python library (Post 2018), and the COMET score[8] (Rei et al. 2022) using the `evaluate` Python library. Furthermore, we used the term recognition component of our pipeline to analyze the number of constraints fulfilled in the output sentences. This is similar to the lemmatized term exact match accuracy (Bergmanis and Pinnis 2021) and exact match accuracy (Alam et al. 2021a), although we do not need to specifically lemmatize the words as our disjunctive constraints include the inflected forms.

For manual evaluation, we generated a spreadsheet that contained one input sentence on each row, the reference translation, and the outputs of each of our tested configurations in random order. We asked a professional translator to evaluate each configuration and mark it either as correct, erroneous due to incorrectly applied constraints (while still present), or erroneous due to other cause (incl. missing constraint) (cf. Bergmanis and Pinnis 2021, [9]). We then calculate percentages of these three categories for each of the evaluated configurations.

[7] `https://github.com/kielikone/mitra-eval-results`

[8] The `wmt22-comet-da` model

[9] Our evaluation differs from that of Bergmanis and Pinnis (2021): They had categories "wrong lexeme" and "wrong inflectional form". We measure the presence of the constraint lexeme automatically, and are more interested in errors caused by incorrect placement of the constraint than those of wrong inflectional form, since the former errors are common in our tests and much more critical.

For the Parliament dataset, the NDFST-based methods (Mitra-FB and Mitra-F) timed out for three sentences, i.e. the beam search never reached the end condition. Similarly, for the Forest Fire Dataset, the trie-based method timed out for one sentence. Those sentences are evaluated as empty strings in the automatic evaluation and left out of the manual evaluation. See Appendix D for an analysis of them.

4.4 Results

The results of the automatic evaluation are presented in Table 5 and the human evaluation in Table 6.

For both datasets, the usage of constraints improved the BLEU, chrF, and TER scores when compared to the unconstrained translations and the general-purpose language model outputs. For the Forest Fire dataset, the BLEU of the unconstrained translations was 16.45, while the BLEU of Mitra-FB was 20.96. For the Parliament dataset, the unconstrained BLEU improved from 34.48 to 35.50 respectively. The COMET score improved from 0.86 to 0.90 with the Forest Fire dataset, while it decreased insignificantly from 0.86 to 0.85 with the Parliament dataset. All automatic evaluation scores for the different Mitra configurations were too near each other to be significant.

Similarly, Mitra-F and Mitra-FB raised the number of fulfilled constraints to 100% from 22.41% for the Forest Fire dataset, with the trie-based methods Mitra-T and Mitra-TB timeouting with one sentence, causing the percentage to drop to 94.83%. For the Parliament dataset, the NDFST-based methods timed out for three sentences, causing the fulfilled constraint percentage to reach only 98.32%, while the unconstrained translations reached 49.46%.

The general-purpose language models achieved BLEU scores comparable to the unconstrained translation with the exception of Mixtral in the English–Finnish direction that produced translations of unusably low quality. Similarly, both Poro and Mixtral fulfilled ca. 82–84% of the constraints with the exception of Poro in the Finnish–English direction. Since Mixtral is arguably better when English is the target language, and Poro when Finnish is the target language, we did not conduct human evaluation for both models, choosing instead the model that performed better for the evaluated language direction.

	Mean time	
Configuration	Failed excl.	Failed = 120 s
Mitra-FB	2.11 s	**2.11 s**
Mitra-TB	1.68 s	3.07 s
Mitra-F	2.36 s	2.36 s
Mitra-T	**1.44 s**	2.83 s
Unconstrained	0.16 s	0.16 s

Table 7: Mean translation times for the Forest Fire datasets. Mixtral and Poro times are not included since they were ran on a different hardware. There was one sentence that timed out with the trie-based configurations. In the first column, that sentence was removed. In the second column, that sentence was given the value equal to the timeout we used, 120 s.

In human evaluation, the NDFST-based approach and backtranslations achieve significantly better results than the Trie and non-backtranslated configurations. For the Forest Fire dataset, they together raise the number of "OK" translations from 52.94% to 60.00%. Similarly, for the Parliament dataset, the number rose from 54.17% to 60.56%. At the same time the number of errors caused by incorrectly applied constraints decreased from 9.41% to 2.35%, and from 17.78% to 10.83%, respectively. For the Forest Fire dataset, the usage of constraints increased the quality when compared to unconstrained translations, while for the Parliament dataset, the unconstrained sentences were evaluated to have higher quality.

For both datasets, the number of "other errors" was considerable. For the Forest Fire dataset, our evaluator noted that the Forest Fire vocabulary did not contain all of the special jargon used in the sentences. Thus, had the vocabulary been more comprehensive, the translation quality could have been better.

4.5 Time performance

The mean translation times for the Forest Fire dataset sentences are presented in Table 7. For constrained translations, the time was measured for the full pipeline, including preprocessing and dependency parsing. The unconstrained translation times are significantly lower than the constrained times, but the time does not include any preprocessing.

Of the sentences which both algorithms were able to translate, the NDFST-based configurations were slower than the trie-based configurations. However, as the trie-based algorithm failed to translate one of the sentences (due to its greediness, it was unable to place the constraints to the sentence correctly, which led to the NMT model considering all hypotheses improbable and never finishing[10]). As the translation timed out, one might argue that the algorithm should be penalized for this by counting the sentence using the timeout as the time it took to "produce" the empty translation. We have reported both numbers in different columns.

We ran Mixtral, Poro, and the Parliament dataset evaluations on different hardware and software environments, so we cannot present comparable numbers for them. We used no batching of multiple sentences.

5 Discussion and Conclusions

In this paper, we have presented *Mitra*, a pipeline for terminologically-constrained machine translation that improves on the previous "hard" constraint methods with a finite state automaton-based constraint recognition algorithm and a backtranslation substitution step. When compared to the trie-based method based on the previously suggested algorithms (Post and Vilar 2018; Hu et al. 2019) without backtranslations, our method significantly increases the quality in human evaluation.

We argue that our method fulfills the three goals we began with: allowing disjunctive constraints, solving the problems of greedy constraint recognition algorithms, and improving quality on "rare and obscure" terms. The finite state automata-based algorithm allows any number of alternatives in the disjunctive constraints, and does not suffer from the greediness of previous algorithms. Furthermore, when combined with backtranslations, the number of errors caused by incorrectly applied constraints drops significantly on both of the evaluation datasets.

While constrained translation significantly improved quality on the Forest Fire dataset, it unexpectedly decreased quality on the Parliament dataset in human evaluation. We believe this discrepancy is due to two main factors. Firstly, the Parliament dataset's vocabulary is not well-suited for constrained generation (see Appendix D for examples), and secondly, the Opus-MT models we used are more familiar with the subject matter, using 49.46% of the constraint terms even when unconstrained, leaving less room for improvement. The Forest Fire dataset, on the contrary, is very unfamiliar to the model, as only 32.94% of the sentences were translated acceptably, and only 22.41% of the desired target terms were used when unconstrained. The effectiveness of constrained translation thus depends on the quality of the constraint vocabulary and the topic of the texts translated.

The major downside of "hard" constraints is their increased time requirement: the translation times were more than ten times larger on three of the four configurations evaluated (see Table 7). Although we did not optimize the evaluation by using batching or other methods such as those recommended by Hu et al. (2019), it is clear that constrained translation is slower in any case. Of our configurations, those that use the NDFST algorithm are slower than those that use tries. However, this is not as major a problem as one might initially think, as in most cases the trie-based solution yields the same result as the NDFST solution. The problem of greediness is only present when two constraints share tokens – if no constraints overlap in this way, there is no ambiguity. Thus, when translating a longer text, the trie-based approach can be used instead of the NDFST solution for most input sentences, making the translation of the whole text nearly as fast as when translated completely with the trie method.

Of the general-purpose large language models evaluated, Mixtral performed very well on the Parliament dataset, producing higher-quality results than constrained decoding methods. Similarly, Poro provides results comparable to the Mitra-T configuration in the human evaluation. Although the percentage of fulfilled constraints is lower than that of constrained translation, ca. 82–84%, the number of "constraint errors" is also low, implying that at least some of the missing constraints can be explained by the models providing a satisfying translation using a synonym or other acceptable construct that does not match the constraint when they have trouble fitting the constraint word into the sentence. Thus, the error mode of LLMs might be considered better than that of constrained decoding-based translation.

Since our pipeline is model-agnostic, it can be used with any NMT model or even with a general-purpose language model. Similarly, many of the "soft" constraint methods can also be combined with our method. We believe that our future research should focus on evaluating these combinations. As even a considerably basic soft method such as backtranslations can improve the translation quality significantly, our hypothesis is that more complex soft methods (such as Bergmanis and Pinnis 2021) can improve it even further.

[10] See Table 1 for a simplified version of the sentence and Appendix D for a full analysis of the failed sentences

Acknowledgments

Both IH and TF are funded by Kielikone Oy. IH is additionally funded by the Doctoral Program in Computer Science at the University of Helsinki. We would like to thank Piia Saresoja for evaluating the translated sentences and Kaarina Hyvönen for lending time towards polishing this text.

References

Alam, Md Mahfuz Ibn, Antonios Anastasopoulos, Laurent Besacier, James Cross, Matthias Gallé, Philipp Koehn, and Vassilina Nikoulina. 2021a. On the evaluation of machine translation for terminology consistency. *Preprint*, arXiv:2106.11891.

Alam, Md Mahfuz Ibn, Ivana Kvapilíková, Antonios Anastasopoulos, Laurent Besacier, Georgiana Dinu, Marcello Federico, Matthias Gallé, Kweonwoo Jung, Philipp Koehn, and Vassilina Nikoulina. 2021b. Findings of the wmt shared task on machine translation using terminologies. In *Proceedings of the Sixth Conference on Machine Translation*, pages 652–663.

Anderson, Peter, Basura Fernando, Mark Johnson, and 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*, pages 936–945, Copenhagen, Denmark. Association for Computational Linguistics.

Arnola, Harri. 1996. Kielikone Finnish-English MT system “TranSmart” in practical use. In *Proceedings of Translating and the Computer 18*.

Beesley, Kenneth R and Lauri Karttunen. 2003. Finite-state morphology: Xerox tools and techniques. *CSLI, Stanford*, pages 359–375.

Bergmanis, Toms and Mārcis Pinnis. 2021. Facilitating terminology translation with target lemma annotations. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3105–3111, Online. Association for Computational Linguistics.

Eduskunta. 2008. Valtioneuvoston termipankki Valter: Eduskuntasanasto.

Ghazvininejad, Marjan, Hila Gonen, and Luke Zettlemoyer. 2023. Dictionary-based phrase-level prompting of large language models for machine translation. *Preprint*, arXiv:2302.07856.

Hasler, Eva, Adrià de Gispert, Gonzalo Iglesias, and 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)*, pages 506–512, New Orleans, Louisiana. Association for Computational Linguistics.

Hokamp, Chris and 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)*, pages 1535–1546, Vancouver, Canada. Association for Computational Linguistics.

Hu, J Edward, Huda Khayrallah, Ryan Culkin, Patrick Xia, Tongfei Chen, Matt Post, and Benjamin Van Durme. 2019. Improved lexically constrained decoding for translation and monolingual rewriting. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 839–850.

Jiang, Albert Q., Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of experts. *Preprint*, arXiv:2401.04088.

Koehn, Philipp. 2009. *Statistical machine translation*. Cambridge University Press.

Koehn, Philipp, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 127–133.

Metsäkeskus. 2022. Tuli metsässä -sanasto. Received: 2023-06-20.

Metsäkeskus. 2023. Tuhonaiheuttajat -sanasto. In prep. Received: 2023-06-02.

Nykänen, Asko. 1996. Design and Implementation of an Environment for Parsing Finnish Sentences.

Post, Matt. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels. Association for Computational Linguistics.

Post, Matt and 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)*, pages 1314–1324.

Qi, Peng, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Rei, Ricardo, José G. C. de Souza, Duarte Alves, Chrysoula Zerva, Ana C Farinha, Taisiya Glushkova, Alon Lavie, Luisa Coheur, and André F. T. Martins. 2022. COMET-22: Unbabel-IST 2022 submission for the metrics shared task. In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 578–585, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Semenov, Kirill, Vilém Zouhar, Tom Kocmi, Dongdong Zhang, Wangchunshu Zhou, and Yuchen Eleanor Jiang. 2023. Updated findings of the wmt 2023 shared task on machine translation with terminologies. In *Proceedings of the Eight Conference on Machine Translation (WMT)*. Association for Computational Linguistics. The updated version available at `https://wmt-terminology-task.github.io/upd_wmt_terminology_2023.pdf`.

SiloAI. 2023. Poro - a family of open models that bring European languages to the frontier. `https://www.silo.ai/blog/poro-a-family-of-open-models-that-bring-european-languages-to-the-frontier`. Accessed: 2024-02-20.

Tiedemann, Jörg. 2020. The Tatoeba Translation Challenge – Realistic data sets for low resource and multilingual MT. In *Proceedings of the Fifth Conference on Machine Translation*, pages 1174–1182, Online. Association for Computational Linguistics.

A Algorithms

We have described our beam search algorithm in detail here for increased reproducibility. Algorithm 1 contains the main loop of the beam search. Algorithm 2 contains the candidate selection algorithm. Finally, Algorithm 3 details the beam allocation algorithm.

Since our algorithm is model-agnostic, we have abstracted the calls to the NMT model by referring to the probability function P and “top-k sampling”. In real implementations, these probabilities would come from the decoder of the NMT model or an LLM model.

```
input  : $M$ maximum length of the output in tokens
         $C$ number of constraint tokens
         $B$ hypothesis bank size
         $S$ beam size
         $V$ the vocabulary
output : best hypothesis

hypotheses ← [[start token]]
cutoff ← $0$
best hypothesis ←null
for $M$ times do
    /* Calculating the new hypotheses                 */
    candidates ←GetCandidates(hypotheses, $V$, $k$)
    hypotheses ←Allocate(candidates, $C$, $B$, $S$)
    /* Updating cutoff                                */
    foreach finished hypothesis $h$ in hypotheses do
        if $P(h) >$ cutoff then
            cutoff ← $P(h)$
            best hypothesis ← $h$
        end
    end
    /* Pruning hypotheses                             */
    foreach hypothesis $h$ in hypotheses do
        if $P(h) \leq$ cutoff then
            remove $h$ from hypotheses
        end
    end
    /* Early stopping                                 */
    if |hypotheses| = $0$ then
        return best hypothesis
    end
end
```

Algorithm 1: The main loop of the beam search.

B Proofs

The “NDFST” in these proofs refers to a non-deterministic finite-state transducer that has the structure described in this paper (see Figure 1). q_0 is the initial state of the NDFST and q_a is the sole accepting state.

Definition. $C(m)$ is the set of constraint end tokens in the output symbol list m. Since the flag for a

```
Function GetCandidates(hypotheses, $V$, $k$) is
    input  : hypotheses from the previous iteration
             $V$ the vocabulary
             $k$ the parameter for top-$k$ sampling
    output : set of candidate hypotheses
    candidates ← $\emptyset$
    /* Constraint continuations from the NDFST
       */
    foreach $h$ in hypotheses do
        if $h$ is not finished then
            foreach token $t$ that would advance the current
              NDFST state do
                append $h + t$ to candidates
            end
        end
    end
    /* Unconstrained candidates                    */
    foreach $h$ in hypotheses do
        append top-$k$ continuations of $h$ to candidates
    end
    return candidates
end
```

Algorithm 2: The candidate selection algorithm. This is a simplified version of the algorithm described in (Post and Vilar 2018, section 3.1), combining their steps 1 and 3.

```
Function Allocate(candidates, $C$, $B$, $S$) is
    input  : the list of candidates
             $C$ number of constraint tokens
             $B$ target hypothesis bank size
             $S$ maximum beam size
    output : a new list of hypotheses
    if $B \cdot (C+1) > S$ then
        return Allocate2(candidates, $C$, $\lfloor \frac{S-1}{C} \rfloor + 1$, $S$)
    else
        return Allocate2(candidates, $C$, $B$, $B \cdot (C+1)$)
    end
end
Function Allocate2(candidates, $C$, $B'$, $S'$) is
    input  : the list of candidates
             $C$ number of constraint tokens
             $B'$ actual hypothesis bank size
             $S'$ actual beam size
    output : a new list of hypotheses
    /* Allocate hypotheses to banks they
       belong based on their number of
       fulfilled constraint tokens                 */
    hypotheses ← []
    foreach $i$ in $C, \ldots, 0$ do
        bank size ← $0$
        foreach candidate $c$ from most probable to least probable
          do
            if bank size $< B'$ then
                if number of constraint tokens in $c \geq i$ then
                    append $c$ to hypotheses
                    remove $c$ from candidates
                    bank size ← bank size $+ 1$
                end
            end
        end
    end
    /* Fill underfilled banks with most
       probable candidates                         */
    foreach $i$ in $C, \ldots, 0$ do
        foreach candidate $c$ from most probable to least probable
          do
            if |hypotheses| $= S'$ then
                return hypotheses
            end
            if number of constraint tokens in $c \geq i$ then
                append $c$ to hypotheses
                remove $c$ from candidates
            end
        end
    end
    return hypotheses
end
```

Algorithm 3: The beam allocation algorithm.

constraint is set in the same transition that generates the constraint end token, $C(m)$ also corresponds to the set of constraints that have their flag set. C is the set of all constraint end tokens possible.

Definition. $S(q, m)$ is the set of strings that the NDFST accepts from the initial state q and the initial output symbol list m.

Theorem. Given a string s partitioned into two parts $s_1 s_2$, so that the NDFST has consumed s_1 but not s_2, and NDFST states (q_0, m_1) and (q_0, m_2), so that $C(m_1) \subsetneq C(m_2)$ and $s_2 \in S(q_0, m_1)$, then $s_2 \in S(q_0, m_2)$.

Proof. The only accepting node in the NDFST is fenced with flag diacritic symbols, each corresponding to a different constraint. Thus, the accepting state can only be reached if all the flags are set, that is, $C(m) = C$.

Since $s_2 \in S(q_0, m_1)$, there must be a path $q_0 \rightarrow q_{i_1} \rightarrow \cdots \rightarrow q_{i_n} \rightarrow q_a$ accepted by the NDFST if (q_0, m_1) is used as an initial state.

As $C(m_1) \subsetneq C(m_2)$, the same path cannot be accepted when (q_0, m_2) is used as the initial state, as for each $c \in C(m_2) \setminus C(m_1)$, the path contains a negative flag diacritic check that prevents the path from being accepted, as the flag for c is already set since $c \in C(m_2)$.

We construct a new path that is accepted when (q_0, m_2) is used as the initial state. For each $c \in C(m_2) \setminus C(m_1)$, we modify the path by replacing each transition beginning from `@D.C`*c*`@:[C`*c*`]` and ending to `@P.C`*c*`.1@:[/C`*c*`]` with $q_0 \rightarrow q_0$. As all of these c were already present in $C(m_2)$, this modification only removes duplicate positive flag diacritic sets. Since all the flags are set in the modified path, it is accepted by the NDFST. QED.

C LLM Prompts

C.1 Finnish–English

```
Please translate the following sentence
using this vocabulary. Respond using
JSON output such as {"translation": "
This is the translation"}.
Vocabulary: joki = river; virtaava vesi
= flowing water; valuma-alue = catchment
 area
Sentence: Vesilaissa joella tarkoitetaan
 virtaavan veden vesistöä, jonka valuma-
alue on vähintään sata neliökilometriä.
```

C.2 English–Finnish

```
Käännä lause suomeksi annetulla
sanastolla. Vastaa JSON-muodossa, esim.
{"käännös": "Tämä on käännös"}.
Sanasto: octopodes = mustekalat; extant
= elävä; subclass = alaluokka;
cephalopod = pääjalkainen; nautilus =
helmivene
Lause: Octopodes are one of the two
extant subclasses of the cephalopods. It
 is also called two-gilled cephalopods.
The other subclass is the nautiluses or
four-gilled cephalopods.
```

D Failed Translations

Several test sentences timed out during the evaluation. This section contains these sentences a well as an analysis of the cause of the error.

To save space, when we list the constraints, we only list the target lemmas, although in reality we give all inflections generated by our phrase inflector module as a disjunctive constraint. The disjunctive constraint also includes differently capitalized versions of the target term, although we list all target terms in lower case here.

D.1 Forest Fires

This evaluation corpus had only one failed sentence, which failed for both trie-based configurations, but not for the NDFST-based configurations.

D.1.1 Sentence 46

Constraints are underlined.

Source sentence: Forest fire regime describes the role of fire in a given area over a given time period and sums up almost all variables related to forest fires: forest fire effects and their influencing factors, forest fire frequency, forest fire severity, forest fire intensity, the size of fire, the time of and reason for ignition, regularity, variation, etc.

Reference translation: Metsäpaloregiimi kuvaa tulen roolia tiettynä aikana tietyllä alueella ja summaa yhteen lähes kaikki metsäpaloihin liittyvät suureet: metsäpalojen vaikutukset ja niihin vaikuttavat tekijät, metsäpalojen toistuvuus, metsäpalon vaikuttavuus, metsäpalon voimakkuus, koko, syttymisajankohta, syttymissyy, säännöllisyys, vaihtelu, jne.

Constraints: {forest fire regime → metsäpaloregiimi}, {forest fire → metsäpalo}, {forest fire → metsäpalo}, {forest fire frequency → metsäpalojen toistuvuus}, {forest fire severity → metsäpalon vaikuttavuus}, {forest fire itensity → metsäpalon voimakkuus}

The timeout of this sentence is caused by two factors: the greediness of the constraint recognition algorithms and the large number of constraints it has. See Table 1 for an example of a simplified version of this sentence that does not timeout since it has less constraints.

The core issue is that after the tokens encoding "Metsäpalo" have been generated, the trie-based algorithm marks the constraint {forest fire → metsäpalo} fulfilled. After this, the beam search generates the tokens encoding "regiimi", but they are not recognized to be a part of the constraint, since the progress of all other constraints was reset when one of the possible constraints tracked simultaneously was marked fulfilled. This means that the constraint {forest fire regime → metsäpaloregiimi} is yet unfulfilled, and the algorithm tries to place it later in the sentence. However, the decoder language model (correctly) considers all those other places to be improbable, and thus the end condition of the beam search is never reached within the time limit.

D.2 Finnish Parliament

This evaluation corpus had two sentences for which the preprocessing pipeline (i.e. morphological analysis and generation) failed due to a bug that we had not time to correct. We left these sentences out of the evaluation. Of the remaining sentences, the NDFST-based configurations failed for three sentences.

These sentences fail due to two main reasons: they contain too many constraints, and the target terms included in the vocabulary are poorly suited to be used as constraints resulting in unnatural translations considered improbable by the decoder language model. Note that while this section only analyses the timed out sentences, the poor suitability of the vocabulary applies also to those sentences that did not time out and is one cause to the poor scores received by the system in human evaluation.

D.2.1 Sentence 150

Constraints are underlined. This sentence was actually a text with two sentences, the second of which was more problematic.

Source sentence: Valtiopäiväasiakirjat julkaistaan painettuina Valtiopäiväasiakirjat-sarjassa sekä nykyään myös eduskunnan sivustolla Internetissä. Valtiopäiväasiakirjat-sarjassa julkaistaan mm. eduskunta-aloitteet, eduskunnan täysistuntojen pöytäkirjat ja niiden ruotsinkieliset lyhennelmät, hallituksen esitykset, valtioneuvoston kirjelmät, tiedonannot ja selonteot, valiokuntien mietinnöt ja lausunnot, eduskunnan vastaukset ja kirjelmät, välikysymykset sekä kirjalliset kysymykset vastauksineen.

Reference translation: Parliamentary documents are published in print form in the series 'Valtiopäiväasiakirjat' and in recent years have been published on Parliament's web pages as well. The series contains, among other documents, parliamentary motions; records of plenary sessions of Parliament and their Swedish summaries; government proposals, communications, statements and reports; committee reports and statements; parliamentary replies and communications; interpellations; and written questions and the replies to them.

Constraints for the first sentence:
{valtiopäiväasiakirja → parliamentary document}, {eduskunta → parliament, finnish parliament, eduskunta}
Constraints for the second sentence: {eduskunta-aloite → parliamentary motion, member of parliament's motion, member's motion}, {eduskunta → parliament, finnish parliament, eduskunta}, {täysistunto → plenary session}, {hallituksen esitys → government proposal}, {valtioneuvoston kirjelmä → government communication}, {valiokunta → committee}, {mietintö → report of the committee, committee report}, {lausunto → statement of the committee, committee statement}, {eduskunnan vastaus → parliamentary reply}, {välikysymys → motion of censure, interpellation}, {kirjallinen kysymys → written question}

The primary reason for the timeout of this text on the NDFST-based configurations is the large number of constraints. The NDFST-based algorithm has exponential time complexity in the worst-case scenario.

This text is also problematic due to the poor suitability of the vocabulary for constrained translation. For example, the phrase "valiokuntien mietinnöt ja lausunnot" ("the reports and statements of the committees") generates the constraints {valiokunta → committee}, {mietintö → report of the committee, committee report}, and {lausunto → statement of the committee, committee statement}, all of which include the word "committee" (valiokunta). Thus the constraints force the beam search hypotheses to contain translations like "the reports of the committee and the statements of the committee of the committees", which the decoder language model obviously considers improbable. In some hypotheses the extraneous "committee" words also appear in completely different (and wrong) places in the translation, causing hallucinations.

D.2.2 Sentence 232

Constraints are underlined.

Source sentence: Eduskunnan tilintarkastajat antavat eduskunnalle kaksi tilintarkastuskertomusta: 1) eduskunnan tilintarkastajien tilintarkastuskertomuksen eduskunnan tilinpäätöksestä, toimintakertomuksesta ja kirjanpidosta sekä hallinnosta ja 2) eduskunnan tilintarkastajien tilintarkastuskertomuksen Valtiontalouden tarkastusviraston tilinpäätöksestä, toimintakertomuksesta ja kirjanpidosta sekä hallinnosta.

Reference translation: The parliamentary auditors submit two reports to Parliament: 1) a report on the financial statements, annual report and accounting, and administration of Parliament; and 2) a report on the financial statements, annual report and accounting, and administration of the National Audit Office of Finland.

Constraints: {eduskunta → parliament, finnish parliament, eduskunta}, {eduskunta → parliament, finnish parliament, eduskunta}, {tilintarkastuskertomus → parliamentary auditors' report, report of the auditors of parliament}, {eduskunta → parliament, finnish parliament, eduskunta}, {tilintarkastuskertomus → parliamentary auditors' report, report of the auditors of parliament}, {eduskunta → parliament, finnish parliament, eduskunta}, {eduskunta → parliament, finnish parliament, eduskunta}, {tilintarkastuskertomus → parliamentary auditors' report, report of the auditors of parliament}, {valtiontalouden tarkastusvirasto → national audit office of finland}

As with the previous sentence, this sentence contains a large number of constraints. However, unlike in the previous case, here most of the constraints are identical. In fact, it only contains three unique constraints. However, all constraints, even if duplicate, will get separate paths in the finite state machine. The algorithm could be easily optimized by adding counters instead of binary flags (cf. Hu et al. 2019). However, the FST library we used did not support them and we did not have time to implement them.

Further issues are caused by the fact that the correct translation of “eduskunnan tilintarkastaja” is “parliamentary auditor”. However, since it is not included in the vocabulary, the only constraint added is {eduskunta → parliament, finnish parliament, eduskunta}. The “parliament” added as a constraint clashes with the correct word “parliamentary” (in our case, both are single tokens, so they don’t share subword tokens), causing hallucinations in some hypotheses as “parliament” is inserted into a wrong place in the sentence, although the best hypothesis in this case includes arguably passable “the auditors of Parliament”.

Again, as in the previous case, the translations given in the vocabulary are unsuitable for constrained translation. The phrase “eduskunnan tilintarkastajien tilintarkastuskertomuksen” generates the constraints {eduskunta → parliament, finnish parliament, eduskunta} and {tilintarkastuskertomus → parliamentary auditors’ report, report of the auditors of parliament}, leading to unnatural translations such as “the parliamentary auditors’ report of the Parliament”. In fact, the literal translation of “eduskunnan tilintarkastajien tilintarkastuskertomus” is “parliamentary auditors’ report”, i.e. the translation given to the last word in the phrase is the translation of the whole phrase. To be suitable for constrained generation, the vocabulary should contain only one-to-one equivalent translations.

D.2.3 Sentence 321

Constraints are underlined.

Source sentence: Jos kansanedustaja kesken vaalikauden kuolee, hänelle myönnetään vapautus tai hänet erotetaan kokonaan edustajantoimestaan tai hän siirtyy Euroopan parlamentin jäseneksi, hänen tilalleen eduskuntaan tulee varaedustaja joko vaalikauden loppuun saakka tai määräajaksi.

Reference translation: If a Member of Parliament dies during the electoral term, is granted a release from office, is dismissed from office, or is elected to the European Parliament, he or she is replaced in Parliament for the remainder of the electoral term or for a specific period of time by a replacement Member.

Constraints: {kansanedustaja → member of parliament, representative, mp}, {vaalikausi → term of parliament, parliamentary term, electoral term}, {edustajantoimi → office of representative, mp’s responsibilities, member’s responsibilities}, {parlamentti → parliament, legislature}, {eduskunta → parliament, finnish parliament, eduskunta}, {varaedustaja → replacement member of parliament, alternate member of parliament, deputy member of parliament}, {vaalikausi → term of parliament, parliamentary term, electoral term},

Again, this sentence has many constraints. Also, like with the previous two cases in the Finnish Parliament dataset, the vocabulary used to generate the constraints is unsuitable for constrained translation. Since the translation of “edustajantoimi” must be either ”office of representative”, “mp’s responsibilities” or “member’s responsibilities”, the phrase “Jos kansanedustaja [. . .] erotetaan kokonaan edustajantoimestaan” (“If a Member of Parliament [. . .] is dismissed from office”) must be translated cumbersomely as “If a Member of Parliament [. . .] is removed from the office of Representative”, where the word “office of Representative” is unnecessarily used instead of just “office”.

Similarly, the translations of the term “varaedustaja” (lit. “replacement representative”) are “replacement member of parliament”, “alternate member of parliament”, or “deputy member of parliament”, all of which contain unnecessarily the word “parliament”. Thus, the phrase “hänen tilalleen eduskuntaan tulee varaedustaja” (“he or she is replaced in Parliament [. . .] by a replacement Member”) must be translated with “he or she is replaced in Parliament by a replacement Member of Parliament”, duplicating the word “Parliament”.

If the constraints force the translator to generate unnatural text, the decoder will again give low scores to all hypotheses, thus making it difficult to reach the end condition in time.