Implementing and Evaluating Multi-source Retrieval-Augmented Translation

Tommi Nieminen and **Jörg Tiedemann** and **Sami Virpioja**
University of Helsinki
{tommi.nieminen,jorg.tiedemann,sami.virpioja}@helsinki.fi

Abstract

In recent years, neural machine translation (NMT) systems have been integrated with external databases with the aim of improving machine translation (MT) quality and enforcing domain-specific terminology and other conventions in the MT output. Most of the work in incorporating external knowledge with NMT has concentrated on integrating a single source of information, usually either a terminology database or a translation memory. However, in real-life translation scenarios, all relevant knowledge sources should be used in parallel. In this article, we evaluate different methods of integrating external knowledge from multiple sources in a single NMT system. In addition to training single models trained to utilize multiple kinds of information, we also ensemble models that have been trained to utilize a single type of information. We evaluate our models against state-of-the-art LLMs using an extensive purpose-built English to Finnish test suite.

1 Introduction

Most NMT systems receive as their input a source sentence on its own, without any additional context. This is problematic, as producing a correct translation often requires information that is external to the source sentence. For instance, source sentences that are translated as part of a larger document have to be consistent with other parts of the document. Even if the translation of a sentence is not constrained by document context, the translation often needs to conform to terminological or phraseological conventions of a genre, domain, or a house style. Beyond acting as a contextual constraint, external information may also improve translation quality by providing the NMT system with translation examples. These examples can simplify the task of translation, as the NMT system does not have to generate the translation from scratch, but can adapt the provided external information.

One method of providing relevant external information to an NMT system is to query an external database for data based on the source sentence, and then either include the retrieved information as part of the NMT system input or constrain the decoding process based on the retrieved information. As similar information retrieval approaches used with large language models are called retrieval-augmented generation (RAG) (Lewis et al., 2020), we will refer to this family of methods as retrieval-augmented translation (RAT), following Hoang et al. (2023).

Even though almost all published RAT methods concentrate on a single kind of retrieved information (usually either terminology or translation memory matches), in actual practical translation scenarios all the kinds of retrieved information are used simultaneously. For instance, a human translator working in a computer-assisted translation (CAT) tool will be provided with matches from both terminology database and translation memories, and they will need to make their translations conform with both of these information sources.

In this article, we introduce several NMT systems, which can utilize different kinds of retrieved information when generating translations. We experiment with both single NMT models that are trained to utilize multiple kinds of information, and with systems that combine models that have been trained to utilize a single kind of information using a novel ensembling method called contrastive ensembling. We also compare our models with instruction-tuned LLMs, which have a native capability of utilizing multiple types of retrieved information.

Our models are trained to utilize two kinds of information:

1. **Fuzzy matches**: Parallel sentences retrieved from a translation database (translation memory or TM). The search is based on the edit distance between the source sentence being trans-

Proceedings of the Tenth Workshop on Machine Translation (WMT), pages 327–339
November 8–9, 2025. ©2025 Association for Computational Linguistics

lated and the source sentence in the translation database. Usually the matches are restricted to those whose normalized edit distance exceeds a specific threshold (in the translation industry a threshold corresponding to a similarity level of 70 percent is normally used).

2. **Term matches**: Terms retrieved from a terminology database (termbase or TB). The database is searched for all the sub-strings of the source sentence being translated, and all terms where the source term matches the sub-string are returned. As the TB usually contains the terms in their dictionary forms, the sub-strings are lemmatized or stemmed before the search.

The motivation for including these two kinds of information is that they are routinely used in the CAT tools that professional translators use. They represent the types of information that are readily available and have been found useful in real-life translation workflows (Hutchins, 1998). Building the RAT system around widely used types of information also ensures that it can easily be integrated into existing workflows.

As mentioned, RAT can be used in two ways: either as constraining the MT system to utilize the retrieved information in its output (especially in the case of terminology), or as providing contextual information to enable the generation of better translations. In this article, we are mainly concerned with constraining RAT, as it has applications in professional translation. One of the problems in developing constraining RAT systems is that the common MT evaluation methods, such as BLEU and COMET, are not well suited to evaluating them, as they provide no information on how well the retrieved information has been utilized. To help us develop and evaluate our models, we therefore compiled an extensive test suite, which contains test cases consisting of source sentences, terms, fuzzy matches, and tests that can be used to check whether the terms and fuzzy matches are used in translations.

2 Related work

We structure our system around retrieval methods that have been used in professional translation since the 1960s. These methods have been developed gradually and organically within the translation industry, so their origins are often unclear. For background on fuzzy match and term retrieval, see Hutchins (1998).

The first MT method that can be characterized as RAT was example-based machine translation (EBMT) (Nagao, 1984), where translations were generated based on examples retrieved from a translation database. Statistical machine translation (SMT) methods could also be characterized as a form of RAT, as they rely on retrieving partial translations from a database of translation fragments, and retrieval was also more explicitly integrated into SMT systems (Koehn and Senellart, 2010).

In the context of NMT, one of the first methods recognizable as RAT was introduced in Gu et al. (2017), where fuzzy matches were retrieved from a TM and the attention component of the model was modified to cover the matches in addition to the source sentence. Constraining NMT to adhere to retrieved terminology was first introduced in Hokamp and Liu (2017), where the beam search decoder is modified to always produce the specified target terms in the output. Song et al. (2019) was the first to implement RAT using data-based methods, by replacing sub-strings in the source sentences of the training data with equivalent target language sub-strings. Dinu et al. (2019) introduced data-based RAT for terminology and Bulte and Tezcan (2019) for fuzzy matches.

While most work on RAT has concentrated on a single kind of information, there has been some recent work on unified RAT, where NMT systems can utilize multiple kinds of information. Wang et al. (2023) prefix the inputs and outputs of their model with three different kinds of retrieved information: fuzzy matches, translation templates, and terms. Raunak et al. (2024) fine-tune an NMT model with data augmented with many different kinds of instructions, some of which are similar to RAT, such as an instruction to utilize a particular term in the translation. Moslem et al. (2023) experiment with prompting an LLM with both terms and fuzzy matches to generate adapted translations. What differentiates our approach from the previous implementations is that we focus exclusively on the types of information that are routinely used in CAT tools, which enables easy integration with professional workflows. We also use an ensemble of specialized models in addition to a single model trained to utilize multiple types of information.

The concept of generating translations using multiple different inputs originates from the field

of multi-source translation (Och and Ney, 2001), where the different inputs are equivalent source sentences in different languages. Firat et al. (2016) were the first to implement multi-source translation by ensembling different NMT models that are provided with different inputs, which resembles our ensembling method.

For evaluation, we will use a dedicated test suite for the phenomena we want to tackle. Test suites have been used for evaluating MT quality since at least the 1990s (King and Falkedal, 1990), and they have become more popular in recent years (see for example Macketanz et al., 2022), as the dramatically improving MT quality has created A demand for more granular evaluation methods. As far as we know, our test suite is the first suite designed specifically for RAT evaluation.

3 Models

We train a selection of models, including separate term and fuzzy match RAT models, and unified RAT models, which process both types of retrieved information. For term models we train models that can process a single term, and models which can process up to ten terms. For fuzzy models, we train models that can process a single fuzzy match, and models that can process up to three fuzzy matches.

Models are created with continued training using the Tatoeba-Challenge (Tiedemann, 2020) models as the base models. As our test suite is English to Finnish, we only train models in that language direction. For all models but one we use the standard transformer model *opusTCv20210807+bt-2021-09-01* as the base model. To see the effect of model size, we also train one model using a base model with the transformer-big architecture (*opusTCv20210807+news+bt_transformer-big_2023-04-13*). Continued training has many advantages compared to training the models from scratch: as continued training is much faster, it is easier to test different model variations and the carbon footprint of the training is smaller. The base models can also be used as strong baselines for evaluation, as they have been trained on all the available data from the OPUS corpus (Tiedemann, 2009).

Our models use special symbols to separate the retrieved information from normal source text (see the left column in Figure 1 for an example of how the symbols are used). The vocabularies of the base models do not have any spare symbols that can be used as these special symbols, so we need to re-purpose some of the existing symbols. We pick ten of the least common symbols from the vocabulary, and assign them as our special symbols. Not all symbols are used in the experiments, but we reserve extra symbols in case more are needed in future experiments with the same models. As the vocabularies remain otherwise identical, we can easily ensemble the trained models with each other and with the base model.

Training is continued with a high-quality subset of the Tatoeba-Challenge dataset that was originally used to train the base models. Ten million sentences are included in the continued training subset. The subset does not include data from crawled corpora due to quality problems associated with them (Kreutzer et al., 2022). The data is also scored with BiCleaner-AI (Zaragoza-Bernabeu et al., 2022), and sentence pairs scoring less than 0.7 are excluded from the subset. The duration of continued training is one epoch, and the learning rate is set to 0.00001 to prevent catastrophic forgetting (McCloskey and Cohen, 1989).

For both fuzzy and term models, the training set is annotated with the appropriate RAT data for that type (see Figure 1 for examples of the annotations). The models are trained using the Marian NMT framework (Junczys-Dowmunt et al., 2018).

3.1 Term models

The terminology models are trained using the data augmentation method first introduced in Dinu et al. (2019): source terms are identified in the source sentence, and target terms are appended to the source sentence after the corresponding source terms. Following Bergmanis and Pinnis (2021) we append the source sentence with lemma forms of the target terms instead of the surface forms, in order to train the model to inflect the provided target terms on the target side instead of copying them directly. We also follow Bergmanis and Pinnis (2021) in using synthetic terms, which are generated by aligning the parallel data on the token-level using *fast-align* (Dyer et al., 2013) and then selecting aligned noun and verb phrases as the synthetic terms. *Stanza* (Qi et al., 2020) was used for lemmatization and to identify noun and verb phrases.

3.2 Fuzzy match models

The fuzzy match models are trained using the Neural Fuzzy Repair (NFR) method introduced in Bulte and Tezcan (2019). We use the continued training

subset as a translation database from which fuzzies are retrieved. The database is searched for matches using the *fuzzy-match*[1] library, and the target sides of the matches are prefixed to the source sentences to produce the training data.

Preparing training data for fuzzy match models is more complicated than for term models. In the term model training data we can always make sure that the target term appended to the source sentence is actually present on the target side, but the situation is different with fuzzy matches. Fuzzy matches have to be retrieved from naturally occurring data, as producing them synthetically is not feasible. Also, fuzzy matches are not binding in the sense that terminology is: in the case of terms, the target term can almost always be used in a translation, but it is very common to have a fuzzy match that cannot be used in any valid translation for a source sentence. Because of this, the ideal training data for fuzzy match models consists of the following types of sentence pairs:

1. **Positive examples**: Sentence pairs, where the source sentence is appended with a usable fuzzy match (i.e. a fuzzy match that can be used in a valid translation for the source sentence), and parts of that fuzzy match are present in the target sentence.

2. **Negative examples**: Sentence pairs, where the source sentence is appended with an unusable fuzzy match (i.e. a fuzzy match that cannot be used in a valid translation for the source sentence), and that fuzzy match is not used in the target sentence.

If fuzzy matches are retrieved based on source similarity, the mix of training examples is not optimal, as it will contain many examples where a usable fuzzy match is not used on the target side. On the other hand, if fuzzy matches are retrieved based only on target similarity, the training set will only contain positive examples, and the model will learn to always copy from the fuzzy matches, even when inappropriate. Because of this, we retrieve fuzzies using both source and target similarity, as in Nieminen et al. (2025).

3.3 Unified model

In addition to the separate term and fuzzy models we also train a unified model, which is trained on both two types of data. The training data for this model is generated by merging the training data for the term and fuzzy models. Specifically, we combine the training data of those models, that are trained to process multiple terms or fuzzies, as the unified model also has to process multiple terms and fuzzies.

4 Multisource ensembling

In addition to using a unified model capable of processing both terms and fuzzies, we ensemble models trained to utilize a single kind of information to produce a system that can utilize multiple kinds of information. Each model in the ensemble has its own input, which is prefixed (full matches) or interleaved (terms) with appropriate retrieved information. See Figure 1 for a schematic of the inference pipeline. The ensemble decoder is implemented by modifying the generation functionality in the HuggingFace *Transformers* library[2].

We experiment with the following ensembling methods (see Figure 2 for a visual example):

1. **Naive ensembling**: The next token probabilities of each model in the ensemble are averaged during inference, with equal weight given to each model.

2. **Contrastive ensembling**: Naive ensembling dilutes the effect of the individual models in the ensemble. This is undesirable in our use case, as we want certain models to have more effect than other models at certain phases of generation. For instance, when the next token to be generated is part of the translation for a source term, we want to emphasize the effect of the terminology model that has been provided that specific source term as part of its input. To achieve this, we compare the next token probability distribution for each model to the token probabilities of a contrast model which has not been provided any external information. If a symbol's probability of a model differs significantly from its probability with the base model, the weight of the symbol is boosted in the ensemble (see source code[3] for implementation details). We use the base model from which the RAT models have been trained from as the contrast model.

[1] https://github.com/SYSTRAN/fuzzy-match

[2] https://github.com/huggingface/transformers

[3] https://github.com/Helsinki-NLP/OpusDistillery/blob/modularization/pipeline/hf/multisource_eval.py

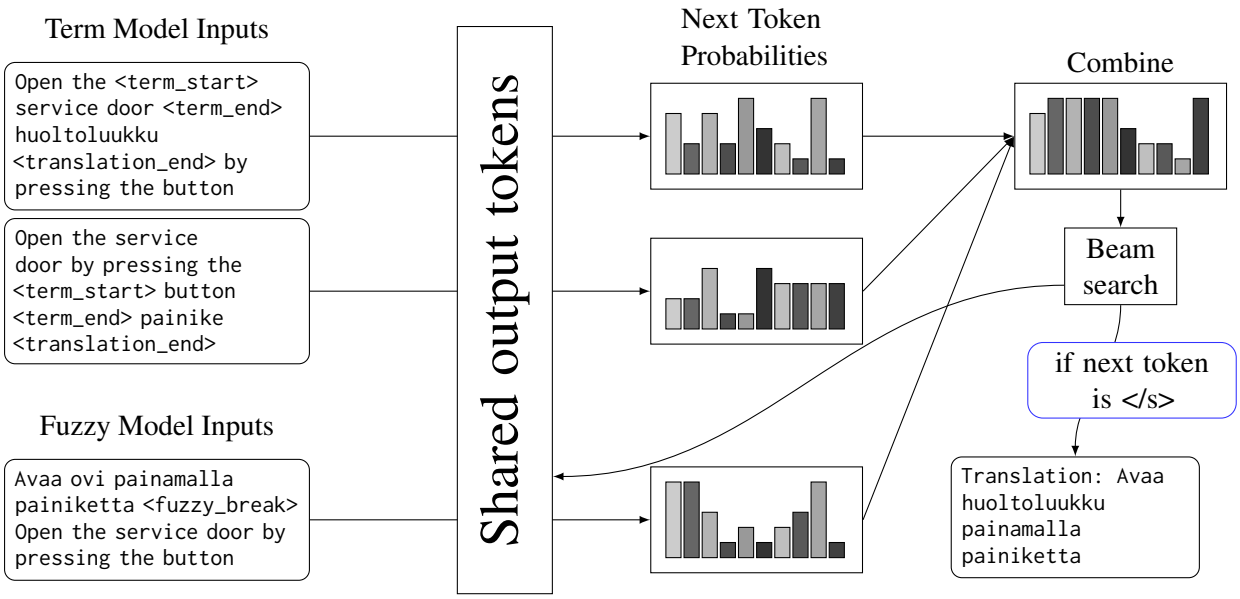


Figure 1: The ensembling inference pipeline with three models and three different inputs. The output tokens are always shared between the models. Note differences in utilization: terms are used completely and inflected, and fuzzy matches are used partially.

Ensembling serves two purposes. First it allows us to utilize multiple kinds of retrieved information during inference, but it also acts as conventional ensembling, the purpose of which is to improve generic output quality. We also hypothesize that ensembling models trained for different RAT methods and provided with different inputs during inference will enhance the quality improving effect of ensembling, as it has been shown (Hoang et al., 2024) that ensembling diverse models produces better results than ensembling similar models, such as different checkpoints of a single training run.

5 Evaluation

RAT systems can be used both for improving generic translation quality and for domain adaptation. When used to improve generic translation quality, it is not relevant whether the translations actually adhere to the terminology and phraseology used in the retrieved examples. For domain adaptation, however, adherence to the retrieved examples is important.

We evaluate model performance from two points of view: correct utilization of the retrieved information in the translations, and general translation quality. For evaluating the correct utilization, we use our test suite, which is covered in detail below. The main purpose of general translation quality evaluation is to see whether continued training with RAT-augmented data degrades generic translation quality. Measuring general translation quality for RAT systems is complicated by the fact that RAT systems are meant to be used with retrieved information: their performance when not provided with any such information is almost irrelevant, as any RAT system can be paired in production with a back-off system that processes source sentences with no retrieved information.

Therefore evaluating RAT systems with standard evaluation sets, which are not paired with retrieved information, does not provide much useful information about the primary use case for RAT systems. Because of these concerns, we use our test suite also as the generic quality test set, so that we can provide retrieved information to the systems. As the test suite does not contain any reference translations, we use the reference-free *wmt23-cometkiwi-da-xl* metric (Rei et al., 2023) for evaluation. This metric has been shown (Freitag et al., 2024) to correlate reasonable well with human judgments and to perform better than many standard reference-based metrics such as BLEU and chrF.

5.1 Test suite

To evaluate the utilization of retrieved information in the generated translations, we created a test suite consisting of source sentences, examples of retrieved information, and tests for checking whether the examples of retrieved information have been utilized in the translations. The test suite was gen-

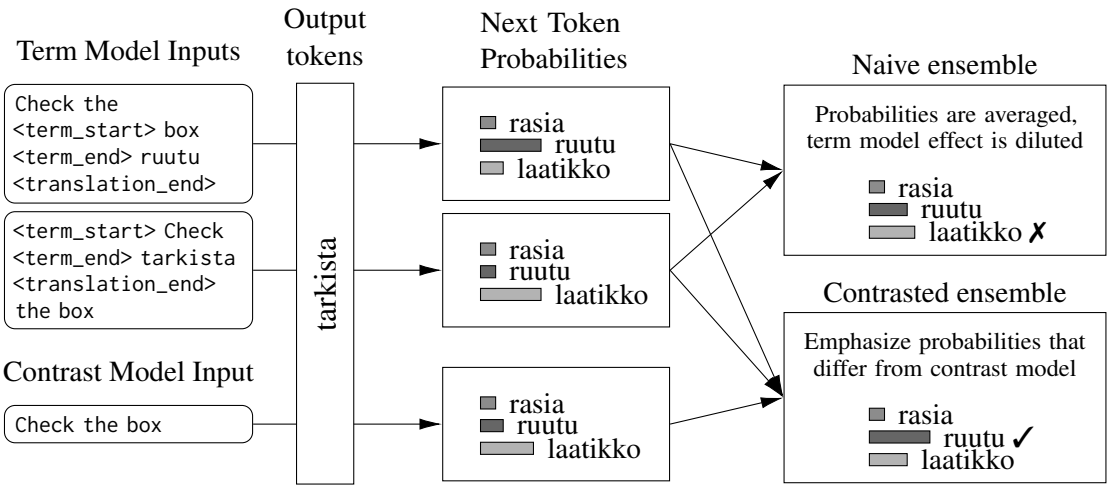


Figure 2: An example of the two ensembling methods. The graph represents a scenario where the source sentence is *Check the box*, and so far the output token *tarkista* has been generated. The word *box* that is being translated in the example is highly polysemous, but the context favours the translation *laatikko*. However, our terminology stipulates that the translation *ruutu* must be used. Naive ensembling produces incorrect output due to treating all models as equal in the context, while contrasted ensembling correctly emphasizes the model that is relevant in the context.

erated semi-automatically: an LLM was used to generate the data, which was then validated and edited by a human reviewer. Initially we tested whether a test suite could be created in a single phase using an LLM (*DeepSeek-V3*), by prompting the LLM to produce complete test cases. While this approach worked for a small test suite (ten or so test cases), when prompted to generate a larger test suite (tens of test cases), the LLM output quality started to degrade noticeably. Because of this degradation, we decided to divide the task into smaller sub-tasks.

As the first step of test suite creation, we prompted the LLM to generate English source sentences. Again, prompting for a large amount of output lead to noticeable output quality degradation, such as repetitive and short sentences. To generate a sufficient amount of high-quality source sentences, we also had to subdivide the sentence generation task to sub-tasks. First we specified seven domains (medical, pharmaceutical, public administration, EU texts, IT administration, IT customer support, and legal), for which sentences could be separately generated for. To add variety, for each of the domains we prompted for the generation of sentences in three different length classes (short, medium, long). In total, we therefore used 21 different prompts, each requesting ten sentences, to generate the source sentences.

The LLM was then prompted to generate a number of fuzzy matches for each of the generated source sentences. For each sentence, three types of fuzzies were generated:

- **Addition fuzzies**: Sentences, which contain additional tokens compared to the source sentence.
- **Deletion fuzzies**: Sentences, which are modifications of the source sentence where some part has been removed.
- **Replacement fuzzies**: Variation of the source sentence where some part has been replaced with a semantically different part.

Note that at this phase only the source side of the fuzzy matches were generated, the target sides were generated later in a separate phase. The edit distance between the generated fuzzies and source sentences was calculated, and all fuzzies below a 70 percent similarity threshold (standard in the translation industry) were automatically discarded.

Once the source sentences and fuzzies had been generated, a human reviewer validated them using a graphical interface (also generated using an LLM) displaying each source sentence and its fuzzy matches on different pages. The reviewer selected 1-5 credible fuzzies for each source sentence. If there were no suitable fuzzies, the reviewer deleted the sentence from the test suite. The reviewer also corrected minor mistakes in some sentences and fuzzy matches.

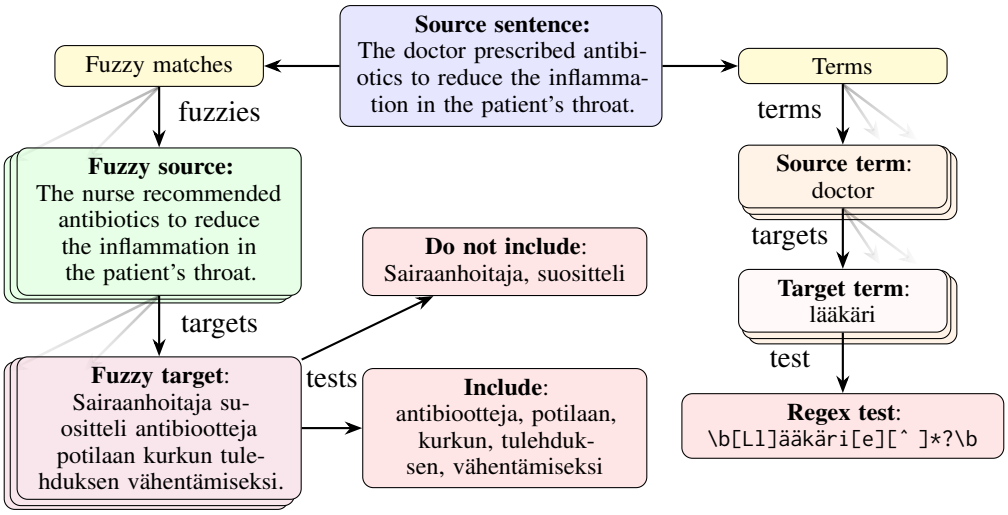


Figure 3: A single test sentence from the test suite. Fading arrows indicate that there can be multiple elements (only one element is shown in the graph to save space).

After the fuzzy validation phase, we prompted the LLM to generate terms and their Finnish translations for each source sentence, as well as regular expression tests for checking whether the term is used in Finnish sentence. The prompt specified that there should be multiple plausible translations for each term. This is important, since if a term has only one plausible translation, it cannot be used to make distinctions between MT systems, as most MT systems are likely to include the correct translation in their output. The terms were also validated manually by a human reviewer using an LLM-generated graphical interface, where the reviewer could select the most plausible terms and their translations, and remove test cases for which no plausible terms had been generated. The reviewer also corrected terms using the interface. The regular expression tests for correct term use that the LLM generated were not usable as such due to them not reflecting Finnish morphology, so the reviewer had to manually correct the tests.

The next phase was to generate translations for the fuzzies. For all other generation phases we used *DeepSeek-V3*, but as the quality of its English to Finnish translations was very uneven, we switched to *GPT4.1* model for generating the translations. The reviewer again reviewed, validated and corrected the translations using an LLM-generated user interface.

The last phase of the test suite generation was generating tests for identifying correct usage of fuzzy matches. Unlike naturally occurring fuzzies, all of the fuzzies in the test suite can be used to construct a valid translation for the source sentence. The test suite is also aimed at scenarios, such as professional translation, where using as much of the fuzzy as possible is desirable, in order to ensure consistency with previous translations. Because of this, we decided to use simple lexical tests to identify whether a fuzzy has been correctly used in a translation. We divided the tokens in each fuzzy into two sets: 1. **Include**: those that correspond semantically to tokens in the source sentence, and should be used in the translation and; 2. **Do not include**: those tokens that have no semantic equivalents in the source sentence and should not be used in the translation.

To create the **Include** and **Do not include** sets, we prompted an LLM with the source sentence, fuzzy source, and the translation of the fuzzy, and instructed the LLM to divide the tokens into the two sets. We also tested traditional word alignment methods, but their accuracy turned out to be too low.

The completed test suite contains 128 source sentences, 434 fuzzy source sentences, 620 fuzzy translations, 403 terms, and 870 term translations. In total, over 200,000 different test cases can be constructed by combining the fuzzy source sentences, fuzzy translations and terms in different combinations. To keep the test suite size manageable for testing (especially with larger models), we create a limited test suite by first organizing the test suite into groups based on how many terms and fuzzy matches each test case contains. Then we randomly pick 50 test cases from each group,

whilst making sure to pick a similar amount of test cases from each domain. We exclude test cases with more than three fuzzy matches to simplify the evaluation task. Our limited test suite contains 1150 test cases.

During evaluation, the test suite produces eight different scores:

1. **Term success (TS)**: the target sentence passes the term test.
2. **Term failure (TF)**: the target sentence fails the term test.
3. **Suitable fuzzy token included (FP)**: the target sentence contains a fuzzy token from the **Include** token list.
4. **Suitable fuzzy token not included (FN)**: the target sentence does not contain a fuzzy token from the **Include** token list.
5. **Invalid fuzzy token not included (IN)**: the target sentence does not contain a token from the **Do not include** token list.
6. **Invalid fuzzy token included (IP)**: the target sentence contains a token from the **Do not include** token list, indicating over-copying.
7. **Suitable fuzzy token bigram included (BP)**: the target sentence contains a bigram of fuzzy tokens from the Include token list.
8. **Suitable fuzzy token bigram not included (BN)**: the target sentence does not contain a bigram of fuzzy tokens from the Include token list.

The bigram scores are included to reward using the same order of tokens in the translation as in the fuzzy. If there are multiple fuzzy matches available to the system, the fuzzy match that has the best overall score is used as the basis of the fuzzy match scores.

We report average term and fuzzy scores for each system, which are calculated with the following formulas:

$$TermScore = \frac{TS}{TS + TF} \quad (1)$$

$$FuzzyScore = \frac{1}{3}\left(\frac{FP}{FP + FN} + \frac{IN}{IN + IP} + \frac{BP}{BP + BN}\right) \quad (2)$$

To facilitate comparison between systems, we also produce a composite score using the following formula:

$$CompositeScore = \frac{5 * TermScore + 3 * FuzzyScore}{8} \quad (3)$$

The composite score intentionally emphasizes term accuracy, as using correct terminology is more important than utilizing fuzzies maximally. Also, in many cases a fuzzy will contain a translation for a term, and if that happens to be different from the specified term, using the correct term lowers the fuzzy scores. Emphasizing term scores compensates for that. The composite score for the whole test suite is calculated as the average of the composite scores for individual test cases, to lessen the effect of test cases with many terms and fuzzies on the overall score.

5.2 Comparison to LLMs

LLMs have been shown to produce better translations than traditional NMT models, at least for high-resource language pairs, such as English to Spanish (Kocmi et al., 2024). LLMs can also be used for RAT by modifying the prompt used to generate the translations (Moslem et al., 2023). While LLM superiority has been shown in the field of generic MT, RAT implemented with NMT (NMT-RAT) has not been thoroughly compared to RAT implemented with LLMs (LLM-RAT). Bouthors et al. (2024) compares NMT-RAT with LLM-RAT using 1-3 fuzzy matches, and finds NMT-RAT much better, but the LLM they compare against does not represent the state of the art.

We compare our models against two recent LLMs, Gemma 3 12B and EuroLLM 9B. We choose these models as they are both competitive and relatively small. We do not compare our models against the largest available models, as our models are aimed at professional translation, where latency and the possibility to deploy models locally is important.

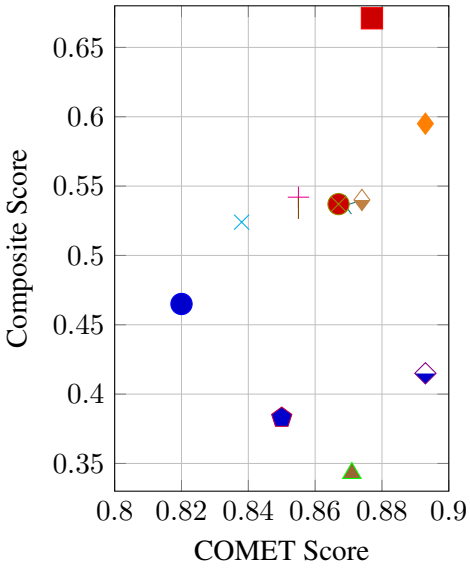
One use case that we foresee for RAT systems is interactive MT, where the MT output is influenced by translator actions. Interactive MT with RAT can for instance take the form of excluding an irrelevant match or modifying a somewhat relevant match manually during translation. This requires very fast generation of translations, as the translator should be able to see the effect of their actions almost immediately. Traditional NMT can gener-

System	Description
Baseline	Standard MT model trained without term or fuzzy annotations.
TermOnly	Model trained with 1-10 term annotations per sentence.
FuzzyOnly	Model trained with 1-3 fuzzy annotations per sentence, using both **source** and **target** similarity fuzzies.
TermAndFuzzy	Model trained with both 1-10 term and 1-3 fuzzy annotations, using both **source** and **target** similarity fuzzies.
TermAndFuzzyBig	Model trained with both 1-10 term and 1-3 fuzzy annotations, using both **source** and **target** similarity fuzzies. Transformer-big model.
ContrastEnsembleTS	Contrastive ensemble of a term and a fuzzy model (trained with both **source** and **target** similarity fuzzies). Each term and fuzzy gets own model in the ensemble.
ContrastEnsembleS	Contrastive ensemble of a term and a fuzzy model (trained with **source** similarity fuzzies). Each term and fuzzy gets own model in the ensemble.
ContrastEnsembleT	Contrastive ensemble of a term and a fuzzy model (trained with **target** similarity fuzzies). Each term and fuzzy gets own model in the ensemble.
BaselineEnsembleTS	Normal ensemble of a term and a fuzzy model (trained with both source and target similarity fuzzies). Each term and fuzzy gets own model in ensemble.
ContrastEnsembleMulti	Contrastive ensemble of a term and a fuzzy model (trained with both source and target similarity fuzzies). Ensemble contains one model for all terms, and one model for all fuzzies.
Gemma3-12B-IT	Instruction-tuned Gemma-3 LLM model with 12B parameters.
EuroLLM-9B-Instruction	Instruction-tuned EuroLLM model with 9B parameters.

Table 1: MT systems evaluated using the test suite.

ate translations fast enough, even running on desktop computers, but it is an open question whether LLMs can achieve the same. While it is not feasible currently to generate translations quickly enough locally with Gemma 3 12B and EuroLLM 9B, we use them as stand-ins for near-future LLMs that can produce translations almost immediately on desktop computers.

5.3 Results



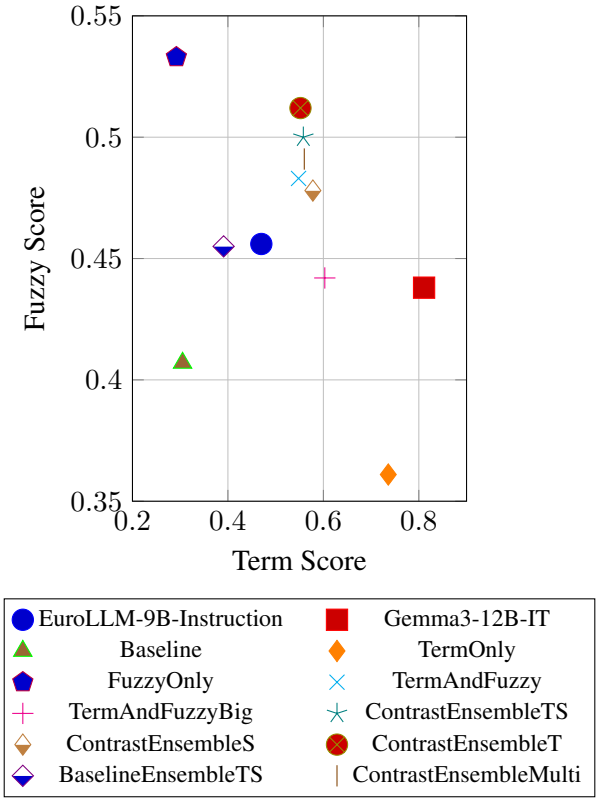


Figure 4: Term score vs Fuzzy score and COMET score vs Composite score.

We evaluated the output of 12 systems with the test suite (see models and their descriptions in Table 1). The main impression of the evaluation is that the

System	COMET ref-free	Comp. score	Term score	Fuzzy score	Fuzzy score (0 terms)
Baseline	0.871	0.343	0.305	0.407	0.520
TermOnly	**0.893**	0.595	0.736	0.361	0.481
FuzzyOnly	0.850	0.383	0.292	**0.533**	0.680
TermAndFuzzy	0.838	0.524	0.548	0.483	0.670
TermAndFuzzyBig	0.855	0.542	0.603	0.442	0.608
ContrastEnsembleTS	0.869	0.536	0.558	0.500	0.**687**
ContrastEnsembleS	0.874	0.540	0.578	0.478	0.642
ContrastEnsembleT	0.867	0.537	0.552	0.512	0.676
BaselineEnsembleTS	**0.893**	0.415	0.391	0.455	0.675
ContrastEnsembleMulti	0.855	0.534	0.560	0.491	0.659
Gemma3-12B-IT	0.877	**0.671**	**0.811**	0.438	0.639
EuroLLM-9B-Instruction	0.820	0.465	0.470	0.456	0.592

Table 2: Test suite scores for each model. The Comet model used is *wmt23-cometkiwi-da-xl*.

test suite is difficult for MT systems, with most systems performing poorly. The strongest performer by far is Gemma-12B-IT, which is also the largest evaluated model. This is a further demonstration of the edge that LLMs have over traditional NMT models, although it should also be noted that the second evaluated LLM (EuroLLM-9B-Instruction) performed worse than the NMT-RAT models. It is also noteworthy that Gemma-12B-IT excels above all in using the correct terminology in its translations. However, it does not score nearly as well in the fuzzy categories. As mentioned, high term accuracy impacts the scores of the fuzzy categories, but Gemma-12B-IT fuzzy score is lower than with the NMT-RAT systems also in cases where there are no terms in the input (see the last column in Table 2).

When comparing the NMT-RAT systems to each other, we can confirm that naive ensembling of models (BaselineEnsembleTS) results in the dilution of the impact of individual models, causing lower term and fuzzy scores. Contrastive ensembling (ContrastEnsemble models) clearly remediates this problem, although term scores remain low compared to the scores produced by the pure term model. The only transformer-big model (TermAndFuzzyBig) in the evaluation has comparable performance to the ContrastEnsemble models, which again demonstrates the effectiveness of contrastive ensembling.

The reference-free COMET scores are fairly similar across systems, with EuroLLM-9B-Instruction being the only outlier. Based on these scores, the RAT methods used do not degrade general output quality.

It is notable that the term scores are low relative to comparable previously published scores, such as those in Alam et al. (2021). This is likely due to the fact that the terms in the test suite have multiple feasible translations by design, which makes the task of applying the correct terminology more difficult.

6 Conclusion

Our experiments demonstrate that combining multiple types of information in a RAT system remains an open problem, even though LLMs show much promise also in this field. The main contributions of this paper are the introduction of contrastive ensembling and the dedicated, extensive test suite for evaluating RAT. Using the test suite we confirm that contrastive ensembling with separate term and fuzzy models provides better results than naive ensembling or single models that can process both terms and fuzzies. While contrastive ensembling does not perform as well as Gemma-12B-IT LLM, its computational requirements are much lower, which makes it suitable to more use cases, such as local low-latency RAT. We have made the training and evaluation pipeline[4] and the test suite[5] available under a permissive license.

7 Limitations

Because we rely entirely on the test suite for evaluation, our experiments are limited to one language

[4]https://github.com/Helsinki-NLP/OpusDistillery/tree/modularization

[5]https://github.com/TommiNieminen/RatTestSuite

direction. However, as our language direction is challenging due to a morphologically complex target language, we can be reasonably confident that the results also apply to less demanding language directions. The test suite has been generated semi-automatically, and while all the test items have been reviewed manually, they may not fully resemble naturally occurring data. The formulas we use to calculate the composite scores of the test suite are motivated by practical considerations, but they may place too much emphasis on certain aspects of the translations, especially correct terminology use. While reference-free MT quality metrics have been shown to work well in recent evaluations (Freitag et al., 2024), they may behave unexpectedly with unusual text types and language pairs.

References

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

Toms Bergmanis 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.

Maxime Bouthors, Josep Maria Crego, and François Yvon. 2024. Retrieving examples from memory for retrieval augmented neural machine translation: A systematic comparison. In *NAACL-HLT*.

Bram Bulte and Arda Tezcan. 2019. Neural fuzzy repair: Integrating fuzzy matches into neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1800–1809, Florence, Italy. Association for Computational Linguistics.

Georgiana Dinu, Prashant Mathur, Marcello Federico, and 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*, pages 3063–3068, Florence, Italy. Association for Computational Linguistics.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648, Atlanta, Georgia. Association for Computational Linguistics.

Orhan Firat, Baskaran Sankaran, Yaser Al-onaizan, Fatos T. Yarman Vural, and Kyunghyun Cho. 2016. Zero-resource translation with multi-lingual neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 268–277, Austin, Texas. Association for Computational Linguistics.

Markus Freitag, Nitika Mathur, Daniel Deutsch, Chikiu Lo, Eleftherios Avramidis, Ricardo Rei, Brian Thompson, Frederic Blain, Tom Kocmi, Jiayi Wang, David Ifeoluwa Adelani, Marianna Buchicchio, Chrysoula Zerva, and Alon Lavie. 2024. Are llms breaking mt metrics? results of the wmt24 metrics shared task. In *Proceedings of the Ninth Conference on Machine Translation*, pages 47–81, Miami, Florida, USA. Association for Computational Linguistics.

Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor O. K. Li. 2017. Search engine guided non-parametric neural machine translation. *ArXiv*, abs/1705.07267.

Cuong Hoang, Devendra Sachan, Prashant Mathur, Brian Thompson, and Marcello Federico. 2023. Improving retrieval augmented neural machine translation by controlling source and fuzzy-match interactions. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 289–295, Dubrovnik, Croatia. Association for Computational Linguistics.

Hieu Hoang, Huda Khayrallah, and Marcin Junczys-Dowmunt. 2024. On-the-fly fusion of large language models and machine translation. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 520–532, Mexico City, Mexico. Association for Computational Linguistics.

Chris Hokamp and Qun Liu. 2017. Lexically constrained decoding for sequence generation using grid beam search. In *Annual Meeting of the Association for Computational Linguistics*.

John Hutchins. 1998. The origins of the translator's workstation. *Machine Translation*, 13:287–307.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. 2018. Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, pages 116–121, Melbourne, Australia. Association for Computational Linguistics.

Margaret King and Kirsten Falkedal. 1990. Using test suites in evaluation of machine translation systems. In *COLING 1990 Volume 2: Papers presented to the 13th International Conference on Computational Linguistics*.

Tom Kocmi, Eleftherios Avramidis, Rachel Bawden, Ondřej Bojar, Anton Dvorkovich, Christian Federmann, Mark Fishel, Markus Freitag, Thamme Gowda, Roman Grundkiewicz, Barry Haddow, Marzena Karpinska, Philipp Koehn, Benjamin Marie, Christof Monz, Kenton Murray, Masaaki Nagata, Martin Popel, Maja Popović, Mariya Shmatova, Steinthór Steingrímsson, and Vilém Zouhar. 2024. Findings of the WMT24 general machine translation shared task: The LLM era is here but MT is not solved yet. In *Proceedings of the Ninth Conference on Machine Translation*, pages 1–46, Miami, Florida, USA. Association for Computational Linguistics.

Philipp Koehn and Jean Senellart. 2010. Convergence of translation memory and statistical machine translation. In *Proceedings of the Second Joint EM+/CNGL Workshop: Bringing MT to the User: Research on Integrating MT in the Translation Industry*, pages 21–32, Denver, Colorado, USA. Association for Machine Translation in the Americas.

Julia Kreutzer, Isaac Caswell, Lisa Wang, Ahsan Wahab, Daan van Esch, Nasanbayar Ulzii-Orshikh, Allahsera Tapo, Nishant Subramani, Artem Sokolov, Claytone Sikasote, Monang Setyawan, Supheakmungkol Sarin, Sokhar Samb, Benoît Sagot, Clara Rivera, Annette Rios, Isabel Papadimitriou, Salomey Osei, Pedro Ortiz Suarez, Iroro Orife, Kelechi Ogueji, Andre Niyongabo Rubungo, Toan Q. Nguyen, Mathias Müller, André Müller, Shamsuddeen Hassan Muhammad, Nanda Muhammad, Ayanda Mnyakeni, Jamshidbek Mirzakhalov, Tapiwanashe Matangira, Colin Leong, Nze Lawson, Sneha Kudugunta, Yacine Jernite, Mathias Jenny, Orhan Firat, Bonaventure F. P. Dossou, Sakhile Dlamini, Nisansa de Silva, Sakine Çabuk Ballı, Stella Biderman, Alessia Battisti, Ahmed Baruwa, Ankur Bapna, Pallavi Baljekar, Israel Abebe Azime, Ayodele Awokoya, Duygu Ataman, Orevaoghene Ahia, Oghenefego Ahia, Sweta Agrawal, and Mofetoluwa Adeyemi. 2022. Quality at a glance: An audit of web-crawled multilingual datasets. *Transactions of the Association for Computational Linguistics*, 10:50–72.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Vivien Macketanz, Eleftherios Avramidis, Aljoscha Burchardt, He Wang, Renlong Ai, Shushen Manakhimova, Ursula Strohriegel, Sebastian Möller, and Hans Uszkoreit. 2022. A linguistically motivated test suite to semi-automatically evaluate German–English machine translation output. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 936–947, Marseille, France. European Language Resources Association.

Michael McCloskey and Neal J. Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. volume 24 of *Psychology of Learning and Motivation*, pages 109–165. Academic Press.

Yasmin Moslem, Rejwanul Haque, John D. Kelleher, and Andy Way. 2023. Adaptive machine translation with large language models. In *Proceedings of the 24th Annual Conference of the European Association for Machine Translation*, pages 227–237, Tampere, Finland. European Association for Machine Translation.

Makoto Nagao. 1984. A framework of a mechanical translation between japanese and english by analogy principle.

Tommi Nieminen, Jörg Tiedemann, and Sami Virpioja. 2025. Incorporating target fuzzy matches into neural fuzzy repair. In *Proceedings of the Joint 25th Nordic Conference on Computational Linguistics and 11th Baltic Conference on Human Language Technologies (NoDaLiDa/Baltic-HLT 2025)*, pages 408–418, Tallinn, Estonia. University of Tartu Library.

Franz Josef Och and Hermann Ney. 2001. Statistical multi-source translation. In *Proceedings of Machine Translation Summit VIII*, Santiago de Compostela, Spain.

Peng Qi, 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*.

Vikas Raunak, Roman Grundkiewicz, and Marcin Junczys-Dowmunt. 2024. On instruction-finetuning neural machine translation models. In *Proceedings of the Ninth Conference on Machine Translation*, pages 1155–1166, Miami, Florida, USA. Association for Computational Linguistics.

Ricardo Rei, Nuno M. Guerreiro, JosÃ© Pombal, Daan van Stigt, Marcos Treviso, Luisa Coheur, José G. C. de Souza, and André Martins. 2023. Scaling up CometKiwi: Unbabel-IST 2023 submission for the quality estimation shared task. In *Proceedings of the Eighth Conference on Machine Translation*, pages 841–848, Singapore. Association for Computational Linguistics.

Kai Song, Yue Zhang, Heng Yu, Weihua Luo, Kun Wang, and Min Zhang. 2019. Code-switching for enhancing NMT with pre-specified translation. 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 449–459, Minneapolis, Minnesota. Association for Computational Linguistics.

Jörg Tiedemann. 2009. *News from OPUS - A Collection of Multilingual Parallel Corpora with Tools and Interfaces*, volume V, pages 237–248.

Jörg Tiedemann. 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.

Ke Wang, Jun Xie, Yuqi Zhang, and Yu Zhao. 2023. Improving neural machine translation by multi-knowledge integration with prompting. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5000–5010, Singapore. Association for Computational Linguistics.

Jaume Zaragoza-Bernabeu, Gema Ramírez-Sánchez, Marta Bañón, and Sergio Ortiz Rojas. 2022. "bicleaner AI: Bicleaner goes neural". In *"Proceedings of the Thirteenth Language Resources and Evaluation Conference"*, pages "824–831", "Marseille, France". "European Language Resources Association".

A Prompt used for generating RAT output with LLMs

Both LLMs tested use the same system prompt, but the user prompt had to be customized for each model to produce clearly delineated translation output.

System prompt for both LLMs: You are a translator translating from English to Finnish.

User prompt for Gemma3-12B-IT: Translate the sentence below to Finnish using the specified terms and fuzzy matches. Use the structure of the fuzzy matches in the translation if appropriate, but do not copy parts of the fuzzy match to the translation if they are not semantically present in the source sentence. Using the specified term is more important than using the fuzzy match, so if a term and the fuzzy match conflict, always prefer the term. Output the answer in the following format, and do not output anything else:
TRANSLATION: TRANSLATION GOES HERE
Terms: *source term 1=target term 1*,*source term 2=target term 1*...
Fuzzy match 1: *target side of fuzzy match 1*
Fuzzy match 2: *target side of fuzzy match 2*...

User prompt for EuroLLM-9B-Instruction: Translate the sentence below to Finnish using the specified terms and fuzzy matches. Use the structure of the fuzzy matches in the translation if appropriate, but do not copy parts of the fuzzy match to the translation if they are not semantically present in the source sentence. Using the specified term is more important than using the fuzzy match, so if a term and the fuzzy match conflict, always prefer the term. Only output the translation.
Terms: *source term 1=target term 1*,*source term 2=target term 1*...
Fuzzy match 1: *target side of fuzzy match 1*
Fuzzy match 2: *target side of fuzzy match 2*...