Analyzing Machine Translation Using KNN Data Stores

Bachelor's Thesis of

Tobias Palzer

Artificial Intelligence for Language Technologies (AI4LT) Lab
Institute for Anthropomatics and Robotics (IAR)
KIT Department of Informatics

Reviewer:	Prof. Dr. Jan Niehues
Second reviewer:	Prof. Dr. Alexander Waibel
Advisor:	M.Sc. Tu Anh Dinh

13th June 2023 – 13th October 2023

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

PLACE, DATE

. . . .

(Tobias Palzer)

Abstract

This study aims to provide an overview over multiple evaluation metrics for neural machine translation using the transformer architecture. Those metrics are all retrieved using similarity search over the generated embeddings of a translation model stored in a datastore. They measure whether similar states were encountered during the training phase, which increases the average translation quality, and how certain the model is in its prediction.

Those metrics can be used both to analyse the model itself, as well as translations created by other models. The effectiveness of these metrics is measured using single-token evaluations and also sequence evaluations. Of this, results of the sequence analysis seem especially promising.

Lastly, the specific requirements to the datastore were analyzed, showing that both a reduced datastore, as well as datastore with a different data base, can be used instead of the datastore consisting of all the embeddings of the training data of the machine learning model.

Zusammenfassung

Diese Arbeit setzt sich zum Ziel, einen Überblick über mehrere Evaluationsmetriken für neuronale Maschinenübersetzung zu bieten. Hierbei wird die “transformer”-Modellarchitektur verwendet. Diese Metriken werden alle über Ähnlichkeitssuche der Einbettungen der Trainingsdaten eines Übersetzungsmodells gewonnen und in einem Datenspeicher abgespeichert. Sie messen, ob ähnliche Zustände während dem Trainieren des Modells existieren, was die durchschnittliche Übersetzungsqualität erhöht, und wie sicher das Modell in seine Vorhersagen ist.

Diese Metriken können sowohl benutzt werden, um das Modell selbst zu analysieren, als auch um Übersetzungen anderer Quellen zu untersuchen. Die Effektivität dieser Metriken wird durch Experimente getestet, die einzelne Tokens und Tokensequenzen betrachten. Die Ergebnisse der Sequenzen sind dabei besonders vielversprechend.

Zuletzt wurden die genauen Anforderungen des Datenspeichers analysiert, wodurch sich gezeigt hat, das sowohl ein reduzierter Datenspeicher, als auch ein Datenspeicher mit anderen Daten als den Trainingsdaten, genutzt werden können.

Contents

Abstract **i**

Zusammenfassung **iii**

1. Introduction **1**

2. Background **3**
2.1. Sequence Generation with Neural Networks 3
2.1.1. Transformer Models for Machine Translation 4
2.2. Translation Evaluation 4
2.3. Nearest Neighbor Datastores and Usage 6

3. Related Work **7**

4. Method **9**
4.1. Nearest Neighbor token store 9
4.2. Evaluation metrics using the KNN-store 9
4.2.1. Obtaining the nearest neighbor metrics 11
4.3. Analyzing single tokens 11
4.4. Analyzing token sequences 12

5. Understanding knn metrics **13**
5.1. Retrieved sentence examples 13
5.2. KNN-metric behaviour 14

6. Experiments and Evaluation **17**
6.1. Setup 17
6.1.1. Model structure 17
6.1.2. Model training and data 17
6.1.3. Test sets 18
6.2. Single Token Experiments 18
6.2.1. Remarks about the used scores 18
6.2.2. Analyzing self generated translations 19
6.2.3. Analyzing premade translations from other sources 21
6.3. Measuring the impact of the decoder layer 23
6.3.1. Results 24
6.4. More fine-grained analysis of errors 24
6.4.1. Errors on different parts of speech 24
6.4.2. Error Severity 25

6.5. Experiments for sequences 26
6.6. Verifying the impact of the datastore 27
6.6.1. Reduced datastore 27
6.6.2. Different datastore 28

7. Conclusion **29**

Bibliography **31**

A. Appendix **35**
A.1. Pre-Labelled dataset comparison 35

List of Figures

2.1. The Transformer Model Architecture 5

5.1. KNN-distance of 3 datasets 15
5.2. Sentence similarity of 3 datasets 15
5.3. Highest likelihood tokens in relation to KNN-data 16

6.1. Occurrence of different error types by distance 20
6.2. Bucket-Analysis of sentence similarity and KNN-distance with self generated translations 21
6.3. Validity of self-generated tokens in relation to retrieved tokens 21
6.4. Threshold analysis for premade translations 22
6.5. Density distribution of the news test set 23
6.6. Validity of premade tokens in relation to retrieved tokens 23
6.7. Average knn-distance categorized by part of speech 25
6.8. KNN-stats with different error severity 26

List of Tables

5.1. KNN retrievals of correct translations 13
5.2. KNN retrievals of incorrect translations 14
5.3. Distribution of knn-distance 14
5.4. Distribution of sentence similarity 14

6.1. Analyzing translations generated by the transformer model 19
6.2. Comparing knn-distance and sentence similarity 20
6.3. Analyzing existing translations 22
6.4. KNN-distance results for each decoder layer 24
6.5. Analysing tokens by part of speech 25
6.6. Correlation of the Comet-score with knn-distance 26
6.7. Using reduced datastores for analysis 27
6.8. Using a different corpus as a datastore 28

1. Introduction

Due to the spread of the global internet, increasingly more people from all over the world are communicating with each other. Despite widespread teaching of English, some people might still prefer to use their native language when communicating or accessing information. This might be because of comfort or lacking knowledge in foreign languages.

Translations created by humans are not sufficient to solve this issue for the following reasons:

1. They are costly

2. They are not directly available and might take some time

3. They might infringe on the privacy of data, e.g. in private emails or messages

Therefore machine translation is used to enable people to freely access information and communicate across national boundaries.

Machine translation was initially based on applied grammar and syntax rules (rule based translation), which systematically translated speech based on individual components. This approach has the following drawbacks: Those rules need to be defined for each language, requiring experts with domain knowledge. Idiomatic language or proverbs would need their own rule, as literal translation usually is not sufficient for a high-quality translation.

Statistical machine translation was the next approach, which collected statistical information about languages from so called bilingual corpa. They then chose those words as a translation, that are most likely to appear based on the information gathered in the corpus. One advantage of this approach is that it can be reused for different languages, with the major requirement being that a bilingual corpus between the language pair exists.

Starting after 2010, neural networks became used extensively for machine translation. Recurrent Neural Networks (RNNs) were primarily used to process sequences by providing memory cells which make neural networks remember state. This model architecture existed before, but was adapted for machine translation [5]. Improvements were made by using a different memory structure (e.g. LSTM [33] or GRU [5]), as well as introducing attention mechanisms [2], which better remember previous states by dynamically looking at encodings of the source sentence.

The transformer model [35] architecture presented an improvement to RNNs. No memory cells are used anymore. Instead the whole sequence of tokens is processed at once using multiple self-attention layers. Transformer models quickly became state of the art in machine translation, as well as other natural language processing tasks, for example large language models like GPT [3].

Despite all these improvements, translation models still make mistakes. Languages are highly complex structures, and translations can be highly context dependent, which requires a deep understanding of the language, or even culture, of the speaker. However, translation models used in isolation do not understand the concept of errors, and will simply generate incorrect tokens.

When generating such a faulty translation, there might still be some hints which indicate errors. In this thesis, we explore how to use embeddings generated by transformer translation models as a measure of similarity to the training data. This is done by first saving the embeddings of the training data in a datastore. When translating sentences using this model, we compare our current state to the saved states from the training data. We measure confidence by similarity to the training data, as well the amount of different choices found in it. For this, we analyze both single tokens as well as token sequences. Next, we compare which metrics are most suitable as a measure for confidence. Lastly, we analyze how dependent this is on both the size and the exact data of the datastore.

2. Background

Machine translation is the task of generating a sequence $(y_1, \ldots, y_m)$ in the target Language T based on a sequence $(x_1, \ldots, x_n)$ in the source language S. In addition to the meaning of the translated sentence, the tone, and feel of a sentence should remain as close to the original sentence as possible.

2.1. Sequence Generation with Neural Networks

As mentioned before, machine translation operates on sequences. Those sequences usually consist of neither words or letters, but tokens. Tokens are subunits of words, ranging from single letters, to whole words. An example of tokenization can be:

$$Lehrerin \xrightarrow{tokenize} Lehrer|in;\ Lehrer \xrightarrow{tokenize} Lehrer$$

One advantage of splitting words in subunits is, that a model can reuse learned associations of a token in multiple contexts, and does not need to learn them anew for each inflection or conjugation. This means that the model can learn to add the suffix “in” to a noun in order to change the grammatical gender of it. If this tokenization was not available, the training data would need to include both forms for every profession in order to let the model make accurate translations, whereas the use of tokenization enables the model to learn this rule.

The first step when building a machine translation model is therefore to tokenize the sentence data. The byte-pair-encoding algorithm [12], originally intended for text/file compression can be adopted for tokenization the following way:

1. Set a target vocabulary size, e.g. 10000.

2. Each character is assigned a token.

3. Find the most common 2-gram of tokens in the corpus, and create a new token for this 2-gram. Replace all occurrences of the 2-gram with the newly created token.

4. If the target vocabulary size has been reached, terminate, otherwise repeat the last step.

The used tokenization algorithm “sentencepiece” [21] builds upon bpe.

2.1.1. Transformer Models for Machine Translation

The transformer neural network structure was first invented specifically to improve machine translation [35], but has found application in other natural language processing applications, most notably the GPT models for general language generation [3]. In contrast to the previous approach, a transformer network does not iterate over the input sentence on a word-by-word basis. Instead, the input layer of the transformer consists of a fixed-size array, which contains all the tokens of the source sentence, as well as an array for the already generated tokens from the target sentence. In addition to the token, a positional encoding is inserted for each token to provide information about word order.

The input layer is followed by multiple encoder and decoder layers. The last layer is a probability distribution over the whole token dictionary. After a forward pass of the model, a token of the output layer is chosen, e.g. the token with the highest probability, and appended to the input layer together with a positional encoding.

In short this means, that a transformer needs only around half the amount of forward passes compared to an RNN, as it does not need to process the input tokens sequentially. Backpropagation now has a fixed cost per token, independent from the sequence length, which decreases training time. In addition, this helps the network to learn "long range dependencies", as the whole sentence and context is always available for each pass, whereas each token in an RNN is only seen once. Therefore, translation of longer sequences are improved.

Each encoder and decoder layer includes "multi head attention", which combines multiple attention heads, which each have individual weights as trained parameters.

$$\begin{aligned} \mathrm{MultiHead}(Q, K, V) &= \mathrm{Concat}\left(\mathrm{head}_1, \ldots, \mathrm{head}_\mathrm{h}\right) W^O \\ \text{where head } &= \mathrm{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right) \end{aligned}$$

In the decoder, Q consists of the previous layer, whereas K and V come from the encoder output. The complete structure of the transformer architecture can be seen in figure 2.1.

2.2. Translation Evaluation

Translation evaluation can be used for multiple purposes: It can serve as a loss-function during the training process of translation models. When used over a large amount of sentence samples, they can be used to evaluate the quality of a model. A third use case is to simply evaluate the translation quality in production settings, for example when translating a large document using machine translation, and trying to find erroneous translations which need to be manually fixed.

When evaluating the quality of translation, two main approaches are possible: The first approach measures the similarity of the generated translation sentence to a given gold translation. An instance of this is the Bleu-score [28], which (in short) works the following way:

The modified n-gram word precision[1], i.e. the percentage of n-word sub-phrases in the hypothesis found in the reference translation, is calculated for multiple values of n. They

[1]Actually a modified precision is used, which penalizes duplicate words in the target translation.

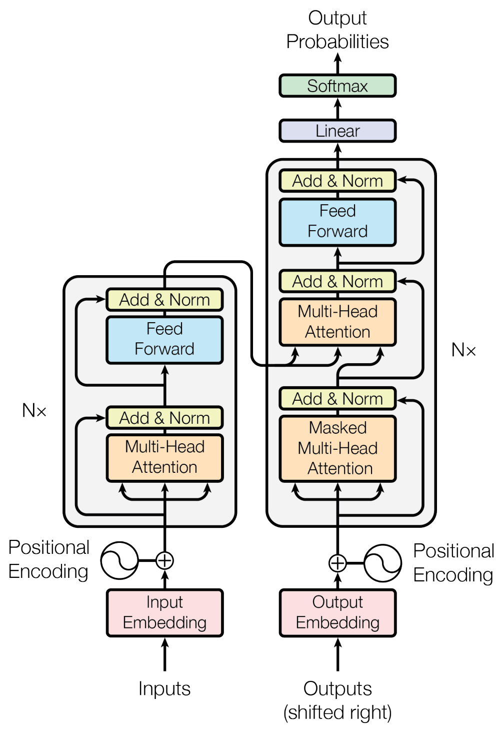

Figure 2.1.: The Transformer Model Architecture
The image is freely provided by Google in their paper [35]

are then averaged using the geometric mean. Lastly a penalty is applied, which punishes shorter translations.

The disadvantage of such an approach is that two correct translations can often still use different words, which results in a lower Bleu-score. This makes it unsuitable for evaluating the translation quality of single sentences. However, the Bleu-score remains a reliable measure for evaluating the quality of whole translation systems, as this error is averaged out over a larger dataset, and it provides high correlation with human judgement [25]. A similar score is the chrF-score [29], where the similarity to a gold translation is determined on the basis of single characters as opposed to complete words.

Quality estimation is a second approach which does not require a gold translation. It usually uses neural networks in order to grade the quality of a translation. Therefore the judgement of a specific translation is usually more accurate compared to scores that rely on a gold translation. Machine translation evaluation metrics based on neural networks outperform lexical metrics in reference to their correlation with human judgment created by human experts [11].

One example of such a score used during the thesis is the COMET-score [30]. It works by first creating cross lingual word embeddings of the hypothesis $\boldsymbol{h}$ and the source $\boldsymbol{s}$ using existing models, e.g. Bert [7]. Those word embeddings are combined in a pooling layer to create a sentence embedding:

$$\boldsymbol{e}_{x_j} = \mu E_{x_j}^{\top} \boldsymbol{\alpha}$$

μ and α are trainable weights, and $E_{x_j}^{\top}$ is a combination of multiple layers of the word embedding. Based on the sentence embedding of the source and the translation the following values are calculated:

$$|\boldsymbol{h} - \boldsymbol{s}|$$
$$\boldsymbol{h} \odot \boldsymbol{s}$$

Those are then appended together with the original data and given to a feed-forward neural network, which assigns a score between 1 and 0 to the translation. Additionally, similar vectors created by a reference translation can also be used to train the model, assuming such a translation is available.

As opposed to evaluating whole sentences, the correctness of individual tokens can also be predicted. This can be done by assigning a confidence score to each token, or using a binary classficator to assign “OK” and “BAD” labels.

2.3. Nearest Neighbor Datastores and Usage

This thesis makes extensive use of Faiss [16], which is a datastore designed to hold a large number of embeddings, generated by neural networks. As opposed to databases, such a vector datastore only contains simple numerical data, and has only limited ways to access them. The standard method to search the datastore is to provide it with a specific embedding, and receive a certain amount of similar embeddings (and their IDs) in return. It is however not guaranteed to return the most similar embeddings. Due to the complexity of the task and the large number of entries, this would not be possible without huge performance costs.

One application for knn datastores is to use them to improve upon text generation [17] or machine translation [18]. This is achieved by first training the model normally. After that, the whole training data is encoded, and the generated embeddings are inserted into the datastore. In this case, embeddings of every token of the sentence are inserted into the datastore, instead of one embedding per sentence. To generate the embeddings, forced decoding is used. In addition to the embedding, the next token from the target sentence, i.e. the expected token during the training process, is also inserted [2].

When translating a sentence, the current embedding is generated by the encoder and decoder. The datastore retrieves similar embedding states, as well as the expected tokens associated with the specific states. The information of which tokens are retrieved is combined with the probability distribution output of the model, and a token is chosen.

It is possible to use any transformer layer, although language models received better results when using the last decoder layer [17]. Instead of the training data, another corpus can also be used as the basis of the datastore, thus making the knn-translation model adapt to new domains, e.g. law, medicine, or IT.

Obviously, the datastore needs to be searched for every generated token, which increases the translation time considerably, compared to only using the transformer model.

[2]The Faiss datastore stores the embedding and the expected token in two different files. Values are linked to each other by assigning them the same index.

3. Related Work

Trusting in the softmax output-probabilities as a measure of confidence is not sufficient when evaluating whether a translated token is correct. To remedy this, multiple approaches have been proposed, which include training machine learning models, and searching the training data for similarity.

[14] proposes the use of the softmax probability of neural networks as a baseline for estimating the confidence in a result for two related problems: The first task is misclassification, where the generated token or assigned label is incorrect. The second task is out-of-distribution detection, which detects whether a sample is part of a distribution that often results in incorrect output of the model.

However [13] has shown classifiers often still have high probabilities even when their results are wrong. Whereas a specific softmax probability in older model architectures generally meant that around the same percentage of samples are predicted correctly, this is not necessarily the case with newer architecture. Concretely, models can overestimate their confidence and assign high probability outputs even on wrong labels.

[36] shows how this overestimated confidence of neural networks is also present in the transformer structure, which is used in the thesis.

[8] trained a neural image classifier by giving it the option to ask for hints in exchange for an additional loss penalty. This trains the model to only ask for hints if it is "confident" in its results. The trained model is then used to estimate the confidence by counting the number of hints used. This approach was adapted for machine translation using transformers by [24]. Detecting out-of-distribution samples corresponds to out-of-domain samples in machine translation. Based on the more accurate confidence measurement, the labels can be smoothed, so that they are more similar to the actual confidence of tokens.

[26] uses the similarity of a sample to the training data of the transformer model as a measurement of confidence. This approach is combined with the output probability for a more comprehensive confidence assessment. Additionally the confidence is also measured for tokens in the input language, which may be useful in certain applications.

[22] provides an alternative way of using the training data of a language model. It uses influence functions to find out which sentences in the training data have the highest influence on erroneous translations filters these sentences from the training data to retrain a better model.

4. Method

The main goal of this thesis is to use an approach inspired by [26] to measure how much information on translation confidence can be gained from similarity data. We introduce different metrics, which will be compared in chapters 5 and 6.

In this section we elaborate on which specific data is used as a measure of confidence when translating sentences, as well as how this data is obtained. After that, we will describe how this data is used, namely for token-level confidence as well as sentence-level confidence.

4.1. Nearest Neighbor token store

For the experiments, the knn-datastore implementation knn-box [38] was used. This toolkit provides a command line interface to build a knn-datastore for an existing model, as well as implementations of translation models which access the knn-datastore. Given that the training data is comparably small, the whole training data can easily be added to the datastore. When using knn-datastores for sequence generation, it is sufficient to only use a part of the training data for the datastore, especially when the data is rather large. This is however mainly for improving the performance by reducing the time searching the datastore. The approach of this thesis is to find similar sentences in the training data. By reducing the size of the datastore, information about certain sentences from the training data is lost, which should weaken the effectiveness of the datastore as a similarity measure. Therefore, the whole training data is put into the datastore, unless specifically mentioned otherwise.

The datastore is accessed by taking the hidden state of a decoding step (e.g. before each token of the hypothesis is generated), and searching for similar hidden states. This returns a list of datastore entries, including the distance from the current hidden state, henceforth referred to as knn-distance. The knn-distance is based on the distance metric built into the Faiss datastore, which is the Euclidean (L2) distance between two embedding vectors p and q by default, defined as:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \tag{4.1}$$

4.2. Evaluation metrics using the KNN-store

We propose using the knn-distance itself as the first quality measurement metric. If only samples with a high knn-distance are retrieved, it suggests that the currently translated sample has few similar states in the training data. Therefore the average translation quality can be expected to be worse than sentences which retrieve samples with low knn-distance.

In addition to the the retrieved token, which is needed for knn-language models, the sentence id of the training sample is also stored in the datastore. This allows for further similarity analysis between the the retrieved training data and the sentence that is to be translated. For instance, we can build sentence embeddings of the most similar source sentences as well as the sentence that will be analyzed.

Our next method of quality measurement is to use cosine similarity between sentence-level embeddings generated by [7][31] as a similarity measure independent from specific translation model encodings and token positions. The cosine similarity is defined as:

$$\cos(x, y) = \frac{\langle x, y \rangle}{|x| \times |y|} \tag{4.2}$$

This metric will from now on be referred to as sentence similarity. The difference between the knn-distance and the sentence similarity is as follows:

The knn-distance is specific to the model that is currently analyzed, whereas the sentence embedding model is independent from the translation model and can be used to analyze multiple translation models. The knn-distance is not supposed to measure the similarity of whole sentences, but of the embeddings generated at specific token positions, which is harder to assign meaning to. As we will show later, two very different sentences might have a very similar embedding in the translation model, as long as they are responsible for generating the same word. When looking at the sentence embedding, which contains their whole meaning, they would have a lower similarity score, as only a small part of the sentence is similar.

It should be noted, that the transformer model has a different embedding for each single output token, whereas the sentence embedding model only has one embedding for the whole input sentence. This measure can therefore be used to check whether sentences, that are seen as similar by the translation model, are actually similar and not just transformed into a similar representation by the translation model.

Other metrics that we derived from the knn data are:

- Number of different knn-proposals: If multiple different tokens are retrieved, it implies a certain “uncertainty”, on which token to use. Of course, multiple different tokens can be correct, but if the most similar states retrieved from the training data all expect the same token, the token generated should more likely be correct

- Model prediction equals retrieved knn-tokens: If the token with the highest possible likelihood as calculated by the translation model corresponds with a higher amount of tokens retrieved from the training data, it is assumed to be more likely to be correct.

These two metrics are restricted by how many nearest neighbors are retrieved from the datastore (in our case 8). The knn distance starts with 0 and has no well defined limit, whereas the sentence similarity ranges from 0 to 1. However there is only a comparatively smaller range of retrieved neighbors, which limits the expressiveness of the last two metrics.

4.2.1. Obtaining the nearest neighbor metrics

We retrieve the aforementioned data by first obtaining a translation. This is either done by taking the gold translation if available, or letting the model generate a translation hypothesis using beam search and taking the hypothesis with the highest likelihood.

Given a translation, another pass over the decoder is required in order to obtain the hidden state of the transformer network after each pass of the decoder. Due to the design of the fairseq library, this is also required for the translations created by the model itself. However this is easily accomplished using forced decoding, which makes the model follow a certain sequence of tokens, and update the inner state accordingly for each step.

Using the hidden states for each token in the sentence, the datastore can now be searched for similar decoder states in the training data. After retrieving them, the data mentioned in the previous section is built and appended to each token. As implied by the name, the knn-datastore retrieves multiple similar states, e.g. the 8 most similar states found heuristically. Depending on circumstances, looking at knn data of multiple states might yield more information than only looking at the most similar state.

It should be noted, that obtaining this data is possible for each layer of the decoder, provided an appropriate datastore was also created. The differences between the layers will be discussed in a later experiment.

4.3. Analyzing single tokens

This thesis explores different applications of knn-data to analyze how they can be used in order to measure the correctness of either single tokens, or token sequences. For single tokens simple annotated data is used, which each token/word being assigned either a “GOOD” label or a “BAD” label. Given that knn-data is token-based, word-based annotations are applied to each token of a word. If the previous metrics can be used to model token confidence, they will show a correlation to the table. For example, tokens with higher knn-distance might be more likely to have a “BAD” label.

It should be noted, that labeling individual tokens is insufficient for comprehensively evaluating a translation. If a word is missing in the translation it cannot be correctly labeled. The binary classification might also be missing a lot of nuance, when grading a token. While technically correct, there are often better word arrangements or expressions. Evaluating whole sequences makes this possible.

For analyzing single tokens, the following methods were used:

- Sorting the tokens by their knn-data (e.g. knn-distance) and putting them in buckets. When using flexible knn-data, e.g. knn-distance or sentence similarity, buckets of equal size are used to provide more accurate data. For knn-data whose value range is the number of retrieved tokens, the number of buckets is fixed
- Measuring their correlation using the Pearson correlation coefficient (PCC)
- Using knn-data as a binary threshold classificator and evaluating it with the Matthews correlation coefficient (MCC) and the F-score
- Kernel density analysis between correct and incorrect tokens

4.4. Analyzing token sequences

The previous methods analyzed how knn-data can be used to analyze whether a single token is correct. For each data point, there is only limited data, and the binary classification also limits the effectiveness of the analysis. When analyzing whole sentences, more data for each instance is available, and different evaluation criteria are more suitable, e.g. a score between 0 and 1.

Each sentence was translated using the transformer model, and assigned a score using the comet [30] framework. The specific models used were “wmt20-comet-qe-da ” and “wmt22-comet-da”. The second model also uses a reference translation in its evaluation.

As a first step, the knn distance of individual tokens is combined using different methods. The resulting value’s Pearson correlation to the comet score is then calculated. We used the following functions to combine the knn-data:

1. Arithmetic mean
2. Median value
3. Arithmetic mean of only the n highest distances
4. Arithmetic mean of only the n lowest distances

All those methods reduce the knn-distance of a whole sequence to a single value. However, advanced methods, such as Recurrent Neural Networks for sequence processing, could also be used to provide more accurate judgements, but are out of scope for this thesis.

5. Understanding knn metrics

After specifying what data is retrieved in which way, we will now further analyze how this data behaves, and how it is normally distributed. Firstly, specific examples of sentences retrieved by the knn-datastore are discussed. This serves to illustrate what type of sentences the neural network considered similar. This is done on a case-by-case basis, and not a statistical analysis. After that, we will present the statistical distribution of this data in 3 sample data sets.

5.1. Retrieved sentence examples

First, an example where the sentence is correctly translated is shown in table 5.1. It can be seen that the retrieved target sentence is only present up to the expected token. The last token of the target in the table is expected by the training data, but has not yet been processed by the model. The source sentence on the other hand, is always processed completely.

In this case, the sentence consists of simple grammar, which results in comparably low knn-distance. With the exception of the first token, only a small amount of unique tokens is retrieved at each step.

Next, an example of a sentence containing an error is found in table 5.2. The error occured at token “called”, which was translated as “genannt”, even though it should be translated as “gerufen” in this context. The first correct knn retrieval, which contains the required token is the one with the eight lowest distance.

Additionally, the retrieval of token “von” instead of “hat” at the second position, has a higher knn-distance and is also incorrect. However, in this case the model correctly chose the token “hat”, even though it is not included in the knn-retrieval.

In both of these cases, the knn-distance is noticeably higher than the rest of the tokens. They also have the highest number of unique retrieved tokens, showing a lower confidence.

Chosen Token	Source retrieval	Target retrieval	KNN-Distance	Unique Retrieved Tokens
Das	This is a degeneration of the retina.	*Das*	53	4
ist	This is a show that’s currently on in Tokyo.	Das *ist*	71	2
eine	This is a photographic representation, called Snapshots.	Das ist *eine*	163	1
Demonstration	... is a demonstration of ...	... ist eine *Demonstration*	324	2
.	This was a perfect introduction.	Dies war eine perfekte Einführung.	133	1

Table 5.1.: KNN retrievals of correct translations
The translated sentence was “This is a demonstration.”.

Chosen Token	Source retrieval	Target retrieval	KNN-Distance	Unique Retrieved Tokens
Wer	Who wrote that	*Wer*	134	1
hat	Any of you who rode the automobiles – was it yesterday? ...	Wer *von*	664	4
Sie	He made you.	Er hat *Sie*	387	1
heute	Here's what some of you did this morning ...	was einige von Ihnen *heute*	421	1
Morgen	... the talk we heard this morning ...	den wir heute *Morgen*	313	3
genannt	What were scientists called before?	Wie wurden Wissenschaftler vorher *genannt*	569	5
?	What were scientists called before?	Wie wurden Wissenschaftler vorher genannt*?*	87	1

Table 5.2.: KNN retrievals of incorrect translations
The translated sentence was "Who called you this morning?".

Type	Average knn-distance	Median knn-distance	Standard deviation
Training Data	262.94	222.81	163.64
Non-Training Data	440.13	414.03	213.45
Random Noise	1084.84	1117.20	156.25

Table 5.3.: Distribution of knn-distance
All values are rounded to the the second decimal. Only the single nearest neighbor is taken into consideration.

5.2. KNN-metric behaviour

To further analyze the behaviour of knn metrics, we analyzed their distribution in 3 cases:

1. Sentences found in the training data of the model, for examples of minimal distances
2. Sentences found in the test data
3. Random noise, i.e. random sequences of tokens, which should be as different from the training data as possible, for examples of maximal distances

For the first two scenarios, 1000 sentences each were used. The third scenario also has 1000 token sequences.

Table 5.3 show some the initial results of the analysis of knn-distance. As expected, the sentences found in the training data have the lowest average and median knn-distance. The sentences not found in the training data have an average distance almost twice as high, and the knn-distance of the random data is the highest by a far margin. Their graphical distribution can also be seen in figure 5.1. It can be seen, that the range of values is widest when analyzing non-training data, as this dataset also has the highest standard deviation.

When looking at the sentence similarity in table 5.4 and figure 5.2, the difference between training data and non-training data becomes even more striking. The relevant sentence

Type	Average cos.-sim.	Median cos.-sim.	Standard deviation
Training Data	0.98	1	0.02
Non-Training Data	0.23	0.20	0.02
Random Noise	0.09	0.08	0.01

Table 5.4.: Distribution of sentence similarity
All values are rounded to the second decimal. Only the single nearest neighbor is taken into consideration.

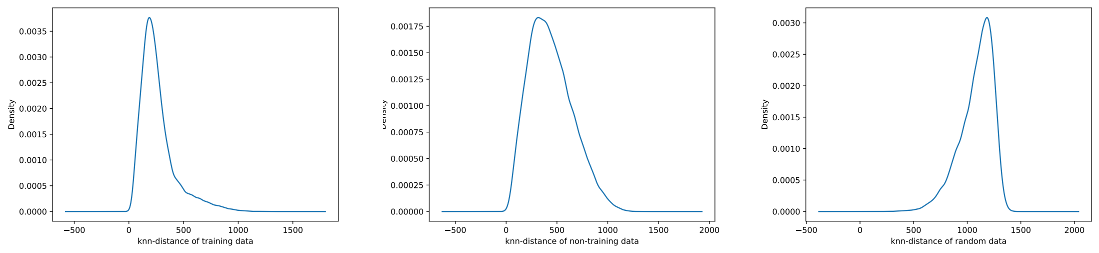

Figure 5.1.: KNN-distance of 3 datasets

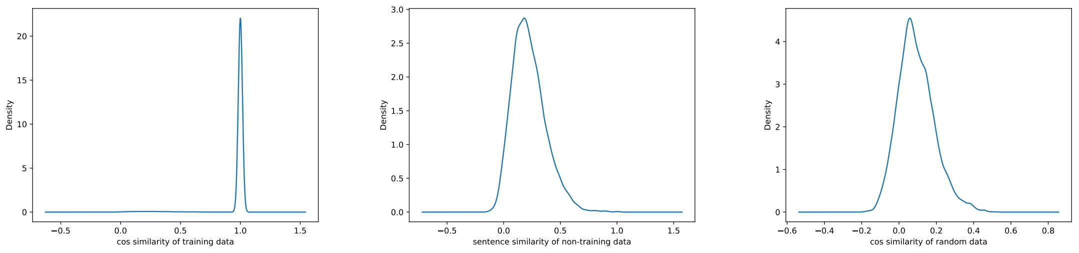

Figure 5.2.: Sentence similarity of 3 datasets

from the training data is almost always found as the most similar neighbor, which leads to an average sentence similarity of 0.98. The knn-distance, as seen in table 5.3, did however not reach values close to zero. This can be because, even though the sentences are found in the training data, they do not exactly generate the same tokens as the reference translation, which results in lower similarity. The sentence similarity instead only compared the source sentences, which were correctly retrieved.

Figure 5.3 also shows the connection between the token with the highest output layer likelihood and the knn-data. It can be seen, that this token is more likely to be chosen when the knn-distance is smaller. Additionally, the higher the distance is, the less likely the token is to be retrieved at least once within the top-{1,5,8} knn retrievals, meaning that there are similar embedding states that expect different tokens, indicating less confidence at higher distances. The same holds true for the sentence similarity in reverse, as higher similarity indicates confidence.

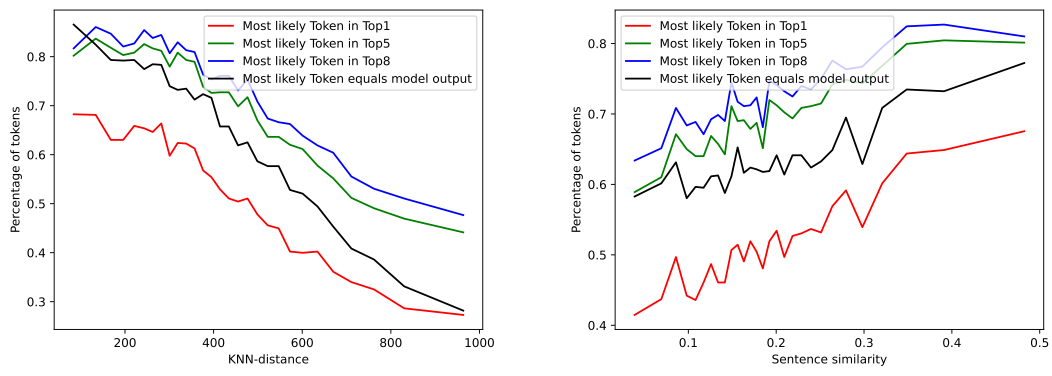

Figure 5.3.: Highest likelihood tokens in relation to KNN-data
Sentences from the second dataset were used. Each data point contains around 800 tokens.

6. Experiments and Evaluation

After giving an overview of the basic behaviour and distribution of knn-data, we will now present how it behaves in relation to translation errors. The first experiments will analyze the general effectiveness for single-token correctness as well as full sentence quality. This analysis is further refined by differentiating between different parts of speech, as well as different types of errors, each of which could show a different degree of influence based on knn-data. Lastly, we will measure how the effectiveness of the method is influenced by changing the datastore as well as the encoder layer from which the knn-data is sourced.

6.1. Setup

6.1.1. Model structure

In order to evaluate the inner workings of transformer translation models, we trained a transformer model to translate from English to the German language. The model architecture is provided by the fairseq modeling toolkit [27], which provides a variety of predefined model architecture architectures and tools. However, as the goal of the thesis is to analyze existing model structures, the standard transformer model by fairseq was used, which is defined as follows:

The encoder first embeds the input array with the dictionary size into a data layer of size 512 32-bit floating point numbers. After that, a positional encoding is added, which is required by transformer structures in order to keep the word order of the sentence. Lastly there are 6 standard encoder layers with self-attention.

The decoder also consists of an embedding layer, followed by 6 decoder transformer layers with self-attention. Lastly, an output layer is added, which transforms the embedding of size 512 into a distribution over the dictionary size, in this case 10112.

This results in a model with a total of 54,493,184 parameters, corresponding to $\approx$ 220 mega-byte. Additionally, the model is trained using dropout [15] and uses the adam optimizer [19] with β values 0.9 and 0.98.

6.1.2. Model training and data

The described model is trained on a corpus of TED talks, which each consist of a person holding a speech on a topic of interest. The sample data was provided during the 11th International Workshop on Spoken Language Translation [4]. The data is formatted by sentences, which each sentence being used as single translation/training step. Before the input can be given to the transformer model, the words are first tokenized, i.e. broken down into subwords, using “sentencepiece”-library by Google [21]. For this a total of 10000 different tokens are allowed. The dataset has a total size of 174443 training sentences for each language pair, which is comparably small. The training lasts for around 3 to 4 hours on one GPU.

6.1.3. Test sets

Depending on the experiment, a number of different test sets were used. These are the following:

TED test data: The test partition of the TED dataset [4] with sentences not used during training, which contains 100 sentences with 2110 labeled tokens.

News test data: An out of domain dataset [1], which contains 100 sentences with 3503 labeled tokens.

We labeled the previous two datasets by hand, differentiating between 8 different kind of errors: Correct, Wrong word, Word order, Ending, Obsolete Word, Proper noun / name, Non-Translation, Incomplete Translation. These labels were inspired by [23] and [34]. However, they do not distinguish between error severity. Additionally, sentences larger than 200 characters where sorted out, in order to make labeling them easier.

Multilingual Quality Estimation and Automatic Post-editing Dataset (QE) [37]: This dataset contains a large amount of translation data, generated by different machine translation models as well as humans. The words are labeled by different human judges. The labels are defined by a character-level interval which specifies the error type and severity. If part of a token is erroneous, the whole token will be considered as such. It is possible for a token to be part of multiple errors in the original dataset. However, we adapted this dataset to only include a maximum of one error-type per token. In the experiments, this was simplified to only include the translations created by a single translation model "Online-A.1574", and the judge "rater4". This results in 708 sentences containing 44153 tokens. Including correct tokens, there are 4 different types of error severities, as well as 8 different types of error (No Error, Fluency, Accuracy, Style, Terminology, Locale-Convention, Non-Translation, Other).

WMT Quality Assessment 2023 (QA): This dataset contains 954 source sentences [9][10], as well as target sentences [1]. This results in a total of 30180 tokens in the target language. Each word has a "Good" or "Bad" label assigned to it.

Europarl data [20]: This corpus was only used to serve as an additional datastore. It contains around 2000000 en-de parallel sentences, of which around 170000 were used.

6.2. Single Token Experiments

6.2.1. Remarks about the used scores

When using the knn-data for a binary classification, a threshold is needed for the MCC and the F-score. In the following results, either the threshold with the highest score is used, or multiple possible threshold values are displayed in a graph. The Pearson correlation

[1]It originally included 1000 sentences, but some sentences had to be filtered out due to conflicts during the tokenization phase.

Dataset	Pearson Correlation	F-Score-	MCC-Score
In-Domain/Ted-Talk, knn-distance	0.21	0.27	0.18
In-Domain/Ted-Talk, Probability	0.07	0.08	0.05
Out-of-Domain/News, knn-distance	0.21	0.36	0.20
Out-of-Domain/News, Probability	0.14	0.12	0.05

Table 6.1.: Analyzing translations generated by the transformer model
For F-score and MCC-score the threshold with the highest value was chosen. All values are rounded to the second decimal value

coefficient works without a threshold, and allows for directly analyzing the correlation between number values (e.g. knn-distance) and the binary classification.

For the F-score, the following is to be kept in mind: It is calculated using the this formula:

$$F = \frac{2}{\text{recall}^{-1} + \text{precision}^{-1}}$$

When using a variable threshold, the F-score has a minimum value > 0. This is because precision of 1 can always be achieved by simply choosing a threshold which includes all incorrect tokens, even if it results in a lower precision. The MCC is therefore a more suitable measurement “which produces a high score only if the prediction obtained good results in all of the four confusion matrix categories (true positives, false negatives, true negatives, and false positives)” [6]. However, for the sake of completeness, the F-score is still included.

The usage of knn data is compared against the baseline of using the last layer of the transformer model, i.e. the output probability.

6.2.2. Analyzing self generated translations

For judging self generated translations, two datasets were used. The first dataset is the TED-talk dataset, providing an in-domain dataset. The second used dataset is the news dataset [1], providing an out of domain dataset.

6.2.2.1. Results

Table 6.1 shows the result of the experiment. It clearly shows an improvement when using the knn-distance as an evaluation metric as opposed to the output probability. Given that the model will chose tokens with a higher probability more often, the usefulness of taking the probability is limited. Therefore even wrong tokens might have a high probability, or they would not have been chosen.

No clear difference between in-domain and out-of-domain data can be seen for the knn-distance metric when looking at the Pearson Correlation and the MCC. The small difference found is more prominent when looking at the probability baseline. The remaining differences when using the knn-distance as a metric can be explained as variance in the datasets.

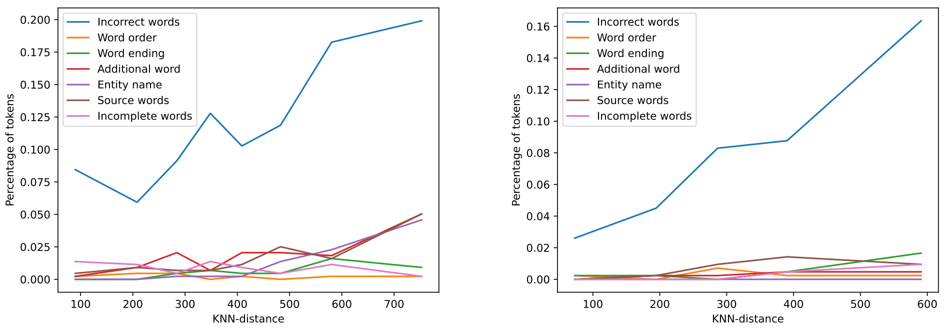

Figure 6.1.: Occurrence of different error types by distance
This figure displays occurrence of errors separated by error type. Each data point represents around 438 different tokens of the news dataset (left) or 422 tokens of the TED dataset (right).

Number of Neighbors	PC of knn-distance	PC of sentence similarity
1	0.210	0.099
2	0.210	0.121
4	0.207	0.126
8	0.205	0.139

Table 6.2.: Comparing knn-distance and sentence similarity
For each token, the knn-distance and sentence similarity of the {1,2,4,8}-nearest neighbors is averaged and used as metric. The Pearson correlation (PC) is rounded to the third decimal.

Figure 6.1 shows how the knn-distance is affected by different types of errors. Sadly due to a big imbalance between the occurrence of different error types little knowledge can be gained. In the news dataset it seems errors besides "incorrect words" react to the knn-distance, but this is less visible in the second dataset. More labeled data with the less common errors might explain which is correct.

Table 6.2 compares the usage of the knn-distance metric against the sentence similarity metric. It also averages multiple metrics of the {1,2,4,8}-nearest neighbors. It can be seen that the knn-distance has correlation of around two times the correlation of the sentence similarity. Interestingly, the knn-distance correlation is almost independent of the amount of nearest neighbors (increasing the values yields only slightly worse results), whereas the sentence similarity sees noticeable improvement when increasing the amount of values. The two metrics are further compared in figure 6.2.

Given that the knn-distance is noticeably better, the experiments in later sections will only make use of this metric, and not the sentence similarity.

Figure 6.3 shows the results of analyzing the number of different retrievals, and the number of retrievals identical to the token chosen by the model. Notably, no clear correlation can be seen in the TED dataset, but there is correlation in the news dataset. This

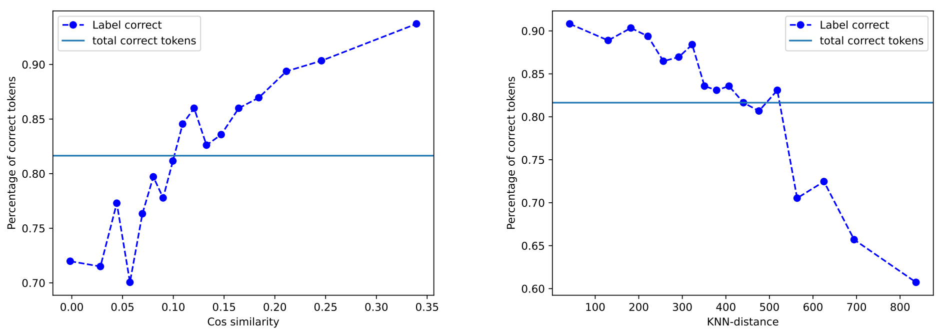

Figure 6.2.: Bucket-Analysis of sentence similarity and KNN-distance with self generated translations
The translations were created with the News data set. Each bucket has around 207 tokens. The y-value consists of the proportion of correct tokens in a bucket.

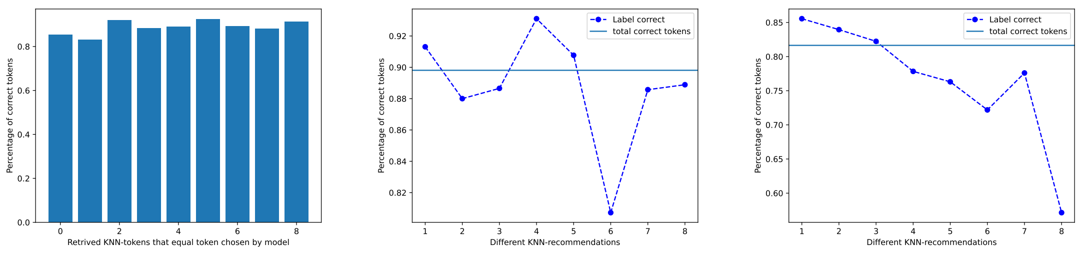

Figure 6.3.: Validity of self-generated tokens in relation to retrieved tokens
The self-labeled TED dataset was used for the first two figures, and the news set dataset for the last image.

might be due to the smaller sample size of the labeled tokens, but also shows that these two metrics are not very robust.

6.2.3. Analyzing premade translations from other sources

Given that teacher forcing is used in order to generate knn-data, it is theoretically possible to generate this data for any sequence of tokens. This can therefore be used to grade existing translations, for example machine generated translations or translations created by human translators. To analyze the feasibility of this approach, two datasets are used:

The QE and QA dataset both include errors split by type. In this experiment, the annotations are interpreted either as correct or incorrect.

6.2.3.1. Results

Figure 6.3 lists the results of this experiment. It can be seen that the results vary greatly between the two datasets. The QA dataset yields results similar to translating the translation generated by the model itself, e.g. having a correlation of around 0.17. However

Dataset	Pearson-Correlation	F-Score	MCC-Score
QA: KNN-Distance	0.17	0.34	0.14
QA: Output-Probability	0.20	0.36	0.18
QE: KNN-Distance	0.07	0.24	0.06
QE: Output Probability	0.11	0.26	0.10

Table 6.3.: Analyzing existing translations
For F-score and MCC-score the threshold with the highest value was chosen. All values are rounded to the second decimal value.

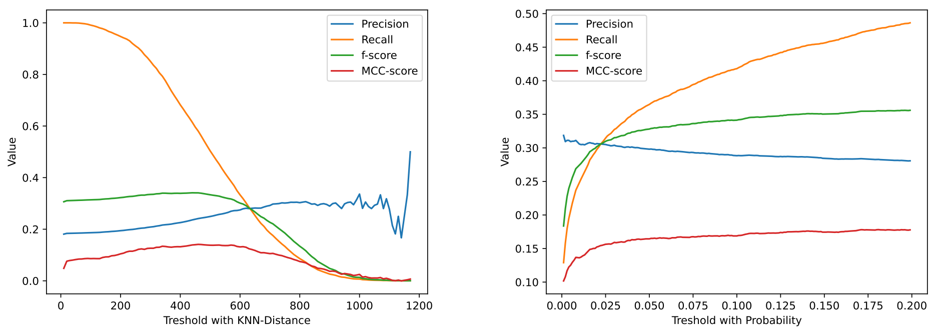

Figure 6.4.: Threshold analysis for premade translations
The QA dataset was used.

the QE dataset does only find a smaller correlation between the knn data-data and the annotations, with the correlation being 0.07. The first dataset suggests that knn-methods can be used to detect errors in other datasets, whereas the second one suggests that the metrics are not refined enough. One possible explanation for this is the different severity of errors found in the two datasets. Upon observation, it seems that the overall translation quality of the sentences found in the second dataset is already quite high. The annotated errors are therefore more of a style or tone mistake of the specific wording. These are labeled as such, but the knn-data does not seem to be sophisticated enough to reliably detect such errors. This shows the limitation of using knn-metrics to analyze translations on a per token basis. Further descriptions of errors found in these datasets are found in Appendix A.

Additionally, the usage of the output probability of the decoder still seems to yield slightly better results than the knn-distance. The results when using the knn-distance as a measure are slightly worse than analyzing translations created by the transformer model itself, which suggests that this method loses some of its effectiveness when forced to emulate different translations, but could also be because of the specific datasets used.

It should be noted, that in this case, both the usage of knn-distance as well as the output probability yield mostly comparable results (using the QE-dataset), whereas analyzing translations created by the model itself showed a stark contrast between these two metrics.

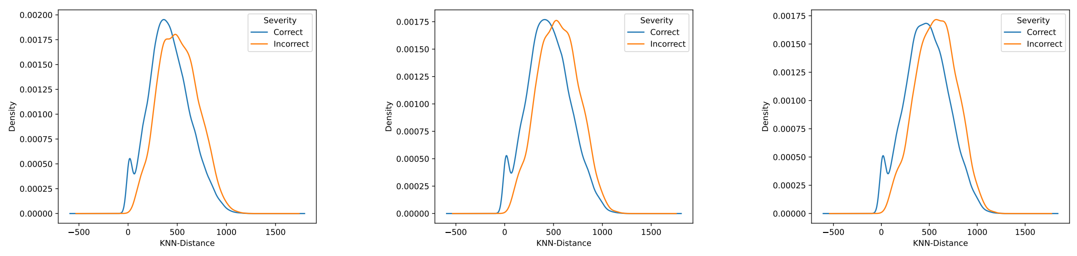

Figure 6.5.: Density distribution of the news test set
This shows the distribution when taking the average of the {1,4,8}-nearest neighbors

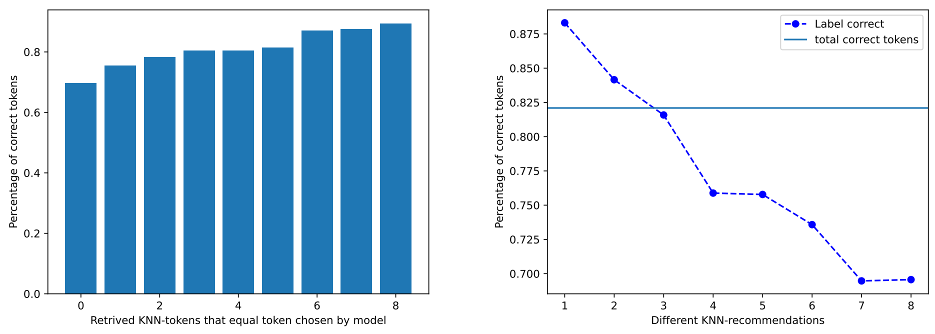

Figure 6.6.: Validity of premade tokens in relation to retrieved tokens
The QA dataset was used to provide the labels.

Figure 6.5 shows the density distribution of correct and incorrect tokens, in dependence on how many next neighbors are used as a reference. It seems that the difference is slightly more pronounced when looking at a higher number of neighbors, although the overlap of the two distributions remains high.

Figure 6.6 shows that the number of different retrievals, and the number of retrievals identical to the token chosen by the model, do show a relation to the validity of the token. This suggests that the results of the self generated tokens are caused by an insufficient sample size and that these two metrics could be part of a confidence measurement, though they might not be very sufficient.

6.3. Measuring the impact of the decoder layer

The previous experiments all used the last layer of the transformer decoder (but not the output layer) as the basis for the knn-datastore. Retrieved decoder states are usually those that expect the same next token, even when the total sentence might have little relevance apart from the expected token.

Therefore, it might be interesting to inspect how the chosen decoder layer impacts the performance of the retrieved metrics. It could be possible that lower levels of the decoder

Decoder Layer	Pearson Correlation	F-Score	MCC-Score
0	<0.01	0.21	0.02
1	0.04	0.30	0.04
2	0.05	0.31	0.05
3	0.07	0.32	0.08
4	0.09	0.32	0.09
5	0.12	0.33	0.12
6	0.17	0.34	0.14

Table 6.4.: KNN-distance results for each decoder layer
The QA dataset was used. All numbers are rounded to the second decimal.

provide a better measure of similarity, as the encoding could be better suited to measure the similarity of the whole sentence, instead of just retrieving encoding states which mostly result in the same output token. For this purpose, a datastore for each decoder layer was generated and analyzed.

6.3.1. Results

Table 6.4 shows the result of the analysis. It can be seen that the effectiveness steadily rises with each layer. The Pearson Correlation additionally has a large jump at the last layer, which is not as pronounced in the F-score and the MCC-score. Therefore, upper decoder levels as basis for the datastore seem to be more suited than lower levels, when evaluating a translation using knn-distance. This is similar to when using knn retrievals for translations, where upper levels also yielded better results [18].

6.4. More fine-grained analysis of errors

6.4.1. Errors on different parts of speech

It might be plausible to assume that certain parts of speech are more directly influenced by knn metrics than others. For example, some parts of speech like pronouns or prepositions fulfill more grammatical roles in a sentence, whereas others are more dependant on specific words, e.g. nouns. Part of Speech labeling was done by the spaCy library [32].

Figure 6.7 shows the average knn-distance listed by part of speech. It seems that the knn-distance for most types of speech is between 400 and 500. The next step is to analyse whether a change in knn-data has a more direct effect on certain parts of speech compared to others.

One problem with this approach is, that it decreases the sample size, as only a reduced number of tokens are available for each individual part of speech. Therefore it is harder to make conclusive judgements about the validity of the data. For this reason the QA dataset is used in this experiment: It provides a large amount of premade tokens, which as shown previously can be analyzed using knn-data.

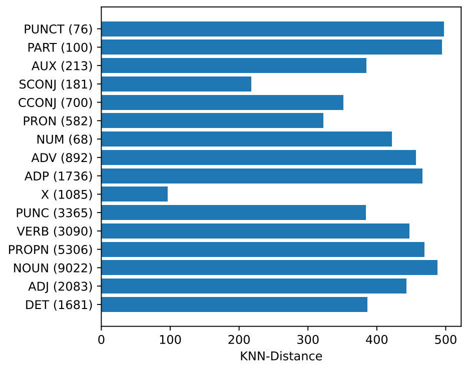

Figure 6.7.: Average knn-distance categorized by part of speech

Part of Speech	Pearson Correlation	MCC
all parts of speech	0.17	0.14
DET	0.09	0.10
ADJ	0.11	0.10
NOUN	0.14	0.13
PROPN	0.17	0.16
VERB	0.09	0.10
PUNC	0.17	0.16
X	0.29	0.30
ADP	0.12	0.11
ADV	0.21	0.18
PRON	0.19	0.17
CONJ	0.13	0.15

Table 6.5.: Analysing tokens by part of speech
The QA dataset was used and only parts of speech with at least 500 tokens were considered.

Table 6.5 shows the results of this analysis. As can be seen, the correlation of the knn-distance to the correctness varies by part of speech.

6.4.2. Error Severity

The QE dataset has annotations separated by error severity: No Error, mild, severe. Given that a higher knn-distance corresponds to a higher likelihood of a token being incorrect, it seems likely that this effect is more pronounced between different categories of error severities. This means, that a severe error would be more likely to have a high knn-distance than a mild error, which in turn is more likely to have a higher distance than a token without any errors.

However, the used dataset previously already demonstrated a low sensitivity for on knn-distance. Therefore, it is unlikely to receive meaningful data in this experiment.

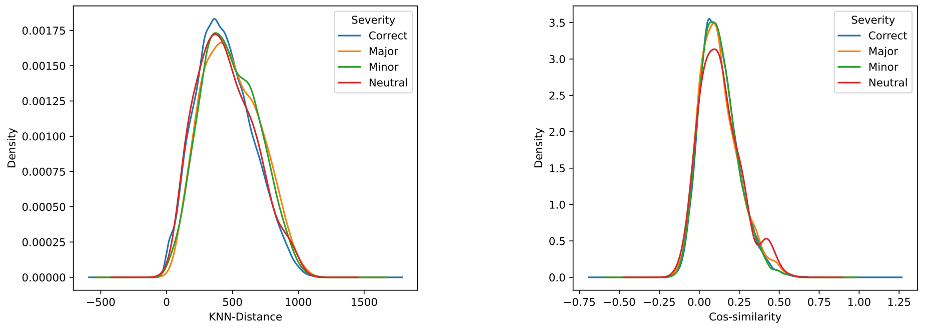

Figure 6.8.: KNN-stats with different error severity
The premade labels from the QE dataset were used.

Test set	Mean	Median	Max 10	Max 5	Max 1	Min 10	Min 5	Min 1
Ted 1 (has target)	0.58	0.52	0.58	0.58	0.53	0.07	0.06	0.01
Ted 2	0.57	0.50	0.56	0.57	0.52	0.07	0.06	0.01
Ted (reduced)	0.45	0.36	0.42	0.44	0.45	0.09	0.05	0.02
News (reduced)	0.45	0.38	0.53	0.50	0.41	0.03	0.14	0.28
QA	0.22	0.20	0.16	0.16	0.15	0.15	0.09	0.05

Table 6.6.: Correlation of the Comet-score with knn-distance
The first four test sets were translations generated by the model, whereas the fifth dataset used forced decoding. Ted 1 also uses a target translation for evaluating the translation quality, whereas Ted 2 and the other test sets only use the source sentence and the translation hypothesis for generating a score. The reduced test sets are the sentences, that were labeled by hand in the previous chapter.

6.4.2.1. Results

Figure 6.8 shows the result of this experiment. Sadly, no relevant differences can be seen in the distribution of the different error severities. This is not surprising, given that any errors in general were almost unable to be detected in this dataset, where the Pearson-Correlation between the knn-distance and the correctness of a token remained only 0.07 (table 6.3). Therefore, this method is not suitable to analyze the error severity, at least in this dataset.

6.5. Experiments for sequences

Next, we analyzed complete sentences by comparing knn-distance to the comet score.

Figure 6.6 lists the results of this experiment. In general, a comparably high correlation is found between the knn-distance and the comet score, at least compared to experiments on single tokens. The best score was delivered by methods 1 and 3. Both of these methods are susceptible to outliers of certain tokens: The arithmetic mean can be greatly influenced by a lower number of high values, and the other metric specifically only samples the highest values found in the sentence.

Datastore Capacity	Pearson Correlation	F-Score-	MCC-Score
100%	0.210	0.273	0.185
80%	0.197	0.270	0.168
60%	0.198	0.279	0.179
40%	0.195	0.279	0.184
20%	0.195	0.265	0.163
10%	0.181	0.266	0.168
5%	0.171	0.262	0.171
2%	0.138	0.248	0.142
1%	0.148	0.260	0.163

Table 6.7.: Using reduced datastores for analysis
The TED dataset was used for this analysis. For all values, knn-distance was used as the metric.

Higher knn-distance corresponds to a lack of similar states generated by the training data. The previous experiments have shown, that this in turn leads to a higher likelihood of erroneous tokens. Thus, the two methods which are most influenced by the higher distance yield the best results. The median, which is resistant to outliers, is therefore a worse metric than the average when trying to find a correlation to the comet score.

Conversely, only looking at the lowest distances yields worse results, even compared to the median knn distance. This can be explained the following way: Incorrect translations also include many correct tokens, which will then be taken as the sole criteria for this metric. On the other hand, correct translations obviously do not include incorrect tokens, which is why using the maximum distance is a better method.

Additionally, the correlation was quite low with the QA dataset. As it was the only dataset analyzed, where the translation was forced and not decided by the model itself, this might suggest that this method is more useful at evaluating translations generated by itself. Overall, the comparably high correlation achieved using such a simple methology seems promising.

6.6. Verifying the impact of the datastore

In this section, the impact of the datastore on the accuracy of the analysis is measured. For this purpose, we exchanged the standard datastore, which contains the whole training data of the translation model, for either a reduced datastore or a datastore with completely different data.

6.6.1. Reduced datastore

When using knn-datastores for translation, the whole training data is not required, given sufficient training data [18]. However, this is a different use case, and does not necessarily hold true for token analysis. For this, the single-token-validity experiment was repeated on different reduced variants of the datastore.

Dataset used in the Datastore	Pearson Correlation	F-Score	MCC-Score
Training dataset, TED talk	0.21	0.273	0.185
Different dataset, Europarl dataset	0.15	0.24	0.14
TED/Training, News	0.21	0.37	0.20
Europarl, News	0.19	0.36	0.20

Table 6.8.: Using a different corpus as a datastore

Table 6.7 shows the result of the experiment. In total, the effect of reducing the datastore was smaller than expected. The Pearson correlation remains largely unchanged between 80% and 20% of the total datastore capacity, with changes only coming into effect at <=10% capacity. This suggests, that only a smaller part of the training data is sufficient for the analysis.

6.6.2. Different datastore

Another way to detect whether similarity between the training data and the currently translated sentence is actually used is to exchange the content of the datastore. As previously mentioned, the knn-datastore can be filled with unrelated translation data, which was not used in the training process, and still works as intended. Therefore, this method might only measure the similarity of a translation to decoder states of correct translations. While such measurements might still be used to judge the quality of a translation, it would be a weaker metric. Ideally, a stronger correlation can be found when using the training data of the model, while a smaller, but still identifiable correlation is found when using other valid translations as a data source.

The dataset of the European Parliament [20] served as substitute data for the datastore. Around the same amount of sentences (170000), as found in the original datastore, was used to build the datastore.

Figure 6.8 shows the results of this experiment. It can be seen, that the Pearson correlation drops from 0.21 to 0.15 in the in-domain TED test dataset. When using the news test set, the reduction is even smaller. This represents a lower reduction in correlation than expected, and suggests that while similarity to the training data can be a factor that increases the usefulness of the method, the effectiveness mostly stems from the similarity to existing, correct sentences.

7. Conclusion

The introduced metrics can be used to a certain extent to analyze errors. This is however heavily dependent on the type of error, and it seems to struggle with high level errors, e.g. the used register, or tone of voice.

Approaches for single tokens, or whole sentences are possible. Single tokens had a comparably low correlation with knn data. This might be, because the task of binary classification is harder, and the presented metrics do not provide enough information to handle this task with a high effectiveness.

Additionally, the knn-distance had a higher correlation to the correctness of a single token, while also needing fewer nearest neighbor look-ups. The other two metrics which counted identical retrieved tokens only showed positive results when using a larger dataset, indicating a lower correlation.

Evaluating whole sequences yielded more promising results. Given a comparably high correlation of knn-distance to existing scoring methods, further work can be done to incorporate this data into quality evaluation systems. For instance, a sequence model could be trained to estimate the quality of a translation based on the sentence as well as its knn-distance per token.

However, the displayed metrics were not influenced that much by the specific datastores. Changing the content or the size of the datastore had only a small influence on the results. Therefore, the used implementation seems to mostly measure how similar a translation is to correct translations, whereas the similarity to the actual training data has a comparably small effect.

The advantage in this is that reduced datasets can be used to analyze a translation, which is especially important for very large training data. Additionally, the training data might not always be available, so using different data as a substitute seems to be an alternative.

Bibliography

[1] URL: `https://data.statmt.org/news-crawl/en/`.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate.* 2016. arXiv: `1409.0473 [cs.CL]`.

[3] Tom B. Brown et al. *Language Models are Few-Shot Learners.* 2020. arXiv: `2005.14165 [cs.CL]`.

[4] Mauro Cettolo et al. "Report on the 11th IWSLT evaluation campaign". In: *Proceedings of the 11th International Workshop on Spoken Language Translation: Evaluation Campaign.* Lake Tahoe, California, Dec. 2014, pp. 2–17. URL: `https://aclanthology.org/2014.iwslt-evaluation.1`.

[5] Kyunghyun Cho et al. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation.* 2014. arXiv: `1406.1078 [cs.CL]`.

[6] Giuseppe Jurman Davide Chicco. "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation". In: *BMC Genomics.* 2020.

[7] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". 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).* Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: `10.18653/v1/N19-1423`. URL: `https://aclanthology.org/N19-1423`.

[8] Terrance DeVries and Graham W. Taylor. *Learning Confidence for Out-of-Distribution Detection in Neural Networks.* 2018. arXiv: `1802.04865 [stat.ML]`.

[9] Marina Fomicheva et al. "MLQE-PE: A Multilingual Quality Estimation and Post-Editing Dataset". In: *arXiv preprint arXiv:2010.04480* (2020).

[10] Marina Fomicheva et al. "Unsupervised Quality Estimation for Neural Machine Translation". In: *Transactions of the Association for Computational Linguistics* 8 (2020), pp. 539–555.

[11] Markus Freitag et al. "Results of WMT22 Metrics Shared Task: Stop Using BLEU – Neural Metrics Are Better and More Robust". In: *Proceedings of the Seventh Conference on Machine Translation (WMT).* Abu Dhabi, United Arab Emirates (Hybrid): Association for Computational Linguistics, Dec. 2022, pp. 46–68. URL: `https://aclanthology.org/2022.wmt-1.2`.

[12] Philip Gage. *A New Algorithm for Data Compression.* 1994. URL: `http://www.pennelynn.com/Documents/CUJ/HTML/94HTML/19940045.HTM`.

[13] Chuan Guo et al. *On Calibration of Modern Neural Networks.* 2017. arXiv: 1706.04599 [cs.LG].

[14] Dan Hendrycks and Kevin Gimpel. "A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks". In: *CoRR* abs/1610.02136 (2016). arXiv: 1610.02136. URL: http://arxiv.org/abs/1610.02136.

[15] Geoffrey E. Hinton et al. *Improving neural networks by preventing co-adaptation of feature detectors.* 2012. arXiv: 1207.0580 [cs.NE].

[16] Jeff Johnson, Matthijs Douze, and Hervé Jégou. *Billion-scale similarity search with GPUs.* 2017. arXiv: 1702.08734 [cs.CV].

[17] Urvashi Khandelwal et al. *Generalization through Memorization: Nearest Neighbor Language Models.* 2020. arXiv: 1911.00172 [cs.CL].

[18] Urvashi Khandelwal et al. *Nearest Neighbor Machine Translation.* 2021. arXiv: 2010.00710 [cs.CL].

[19] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization.* 2017. arXiv: 1412.6980 [cs.LG].

[20] Philipp Koehn. "Europarl: A Parallel Corpus for Statistical Machine Translation". In: *Proceedings of Machine Translation Summit X: Papers.* Phuket, Thailand, Sept. 2005, pp. 79–86. URL: https://aclanthology.org/2005.mtsummit-papers.11.

[21] Taku Kudo and John Richardson. *SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing.* 2018. arXiv: 1808.06226 [cs.CL].

[22] Tsz Kin Lam, Eva Hasler, and Felix Hieber. *Analyzing the Use of Influence Functions for Instance-Specific Data Filtering in Neural Machine Translation.* 2022. arXiv: 2210.13281 [cs.CL].

[23] Peter Lang. "A Classification of Errors in Translation and Revision". In: (2008).

[24] Yu Lu et al. *Learning Confidence for Transformer-based Neural Machine Translation.* 2022. arXiv: 2203.11413 [cs.CL].

[25] Nitika Mathur, Timothy Baldwin, and Trevor Cohn. "Putting Evaluation in Context: Contextual Embeddings Improve Machine Translation Evaluation". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics.* Florence, Italy: Association for Computational Linguistics, July 2019, pp. 2799–2808. DOI: 10.18653/v1/P19-1269. URL: https://aclanthology.org/P19-1269.

[26] Jan Niehues and Ngoc-Quan Pham. "Modeling Confidence in Sequence-to-Sequence Models". In: *Proceedings of the 12th International Conference on Natural Language Generation.* Tokyo, Japan: Association for Computational Linguistics, Oct. 2019, pp. 575–583. DOI: 10.18653/v1/W19-8671. URL: https://aclanthology.org/W19-8671.

[27] Myle Ott et al. “fairseq: A Fast, Extensible Toolkit for Sequence Modeling”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 48–53. DOI: `10.18653/v1/N19-4009`. URL: `https://aclanthology.org/N19-4009`.

[28] Kishore Papineni et al. “Bleu: a Method for Automatic Evaluation of Machine Translation”. In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, July 2002, pp. 311–318. DOI: `10.3115/1073083.1073135`. URL: `https://aclanthology.org/P02-1040`.

[29] Maja Popović. “chrF: character n-gram F-score for automatic MT evaluation”. In: *Proceedings of the Tenth Workshop on Statistical Machine Translation*. Lisbon, Portugal: Association for Computational Linguistics, Sept. 2015, pp. 392–395. DOI: `10.18653/v1/W15-3049`. URL: `https://aclanthology.org/W15-3049`.

[30] Ricardo Rei et al. “COMET: A Neural Framework for MT Evaluation”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 2685–2702. DOI: `10.18653/v1/2020.emnlp-main.213`. URL: `https://aclanthology.org/2020.emnlp-main.213`.

[31] Nils Reimers and Iryna Gurevych. “Making Monolingual Sentence Embeddings Multilingual using Knowledge Distillation”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Nov. 2020. URL: `https://arxiv.org/abs/2004.09813`.

[32] *spaCy: Industrial-strength NLP*. URL: `https://spacy.io/`.

[33] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. *Sequence to Sequence Learning with Neural Networks*. 2014. arXiv: `1409.3215 [cs.CL]`.

[34] Irina Temnikova. “Cognitive Evaluation Approach for a Controlled Language Post-Editing Experiment”. In: *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC’10)*. Valletta, Malta: European Language Resources Association (ELRA), May 2010. URL: `http://www.lrec-conf.org/proceedings/lrec2010/pdf/437_Paper.pdf`.

[35] Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv: `1706.03762 [cs.CL]`.

[36] Shuo Wang et al. “On the Inference Calibration of Neural Machine Translation”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 3070–3079. DOI: `10.18653/v1/2020.acl-main.278`. URL: `https://aclanthology.org/2020.acl-main.278`.

[37] *WMT QE Shared Task 2023*. 2023. URL: `https://wmt-qe-task.github.io/`.

[38] Wenhao Zhu et al. *kNN-BOX: A Unified Framework for Nearest Neighbor Generation*. 2023. arXiv: `2302.13574 [cs.CL]`.

A. Appendix

A.1. Pre-Labelled dataset comparison

In chapter 6 different prelabeled sentences from two different dataset were analyzed. However, the found correlation differed strongly between the two datasets. By inspecting the actual translation errors found in the datasets, we try to provide some context for this phenomenon. First, we take a look at 5 sentences of the QA dataset:

- At Milepost 20, the uphill grade remained at a constant 2.0% ever since before Milepost 4.

  Bei **Milepost** 20 blieb die Steigung **seit Milepost** 4 konstant bei 2,0

- Population densities of eastern grey kangaroos usually peak near 100 per km^2 in suitable habitats of open woodlands.

  Die Bevölkerungsdichte östlicher grauer Kängurus erreicht in geeigneten Lebensräumen offener Wälder in der Regel **fast** 100 pro km ^{2}.

- Vermeil previously led the St. Louis Rams to a victory in Super Bowl XXXIV.

  Vermeil **zuvor führte** die St. Louis Rams zu einem Sieg **in** Super Bowl XXXIV.

- Fort Brown inflicted additional casualties as the withdrawing troops passed by the fort.

  Fort Brown **verursachte** zusätzliche **Opfer**, als die zurückziehenden Truppen an der Festung vorbeizogen.

- Housing is guaranteed for freshmen and sophomores, but not for juniors and seniors.

  Das **Gehäuse** ist für **Neulinge** und **Sophomoren** garantiert, nicht aber für Junioren und Senioren.

Next are some examples of the QE dataset. Errors are once again highlighted, and the specific error type is inserted in brackets after the error.

- we have contacted the family to discuss the situation with them

  und wir haben uns mit der Familie in Verbindung gesetzt, um die Situation **gemeinsam zu besprechen** (style/awkward, minor)

- Lay said the letters all have a message at the bottom urging people to go paperless. "" I'm thinking they had to take down half of a forest just to send this,"" she said.

"Lay sagte, die Briefe hätten alle eine Botschaft am unteren Ende, in der die Menschen aufgefordert werden, papierlos **zu werden** (Style/Awkward, minor). ""Ich denke, sie mussten einen halben Wald **abnehmen** (Accuracy/Mistranslation, major), nur um das zu senden"" (Fluency/Punctuation, minor)

- European nations were so eager to negotiate with Iran that they suggested the US ...

  Europäische Länder strebten derart nach Verhandlungen mit dem Iran, **dass sie vorschlugen, dass** (Style/Awkward, minor) die USA ...

- ""Three out of the six countries, that is the chancellor of Germany, prime minister of Britain, and president of France, all insisted for the meeting to be held, saying that the US would lift all sanctions,"" stated the Iranian president.

  """(Fluency/Punctuation, minor) Drei der sechs Länder, also der **Kanzler** (Accuracy/Mistranslation, minor) **V**on (Fluency/Spelling, minor) Deutschland, Premierminister von Großbritannien und Präsident Frankreichs, haben alle darauf bestanden, dass das Treffen stattfinden sollte, und erklärten, dass die USA alle Sanktionen aufheben würden"", **erklärte** (Style/Awkward) der iranische Präsident."

- Trump quickly denied that he had offered Iran any relief, tweeting that Tehran had asked him to lift sanctions as a prerequisite to talks, but that he had ""of course"" refused.

  Trump verneinte schnell, dass er dem Iran jegliche Entlastungen angeboten habe, und schrieb auf Twitter, dass **der** (Fluency/Grammar, minor) Teheran ihn als Voraussetzung für Gespräche darum gebeten hätte, die Sanktionen aufzuheben, er dies aber „natürlich“ abgelehnt habe.

In this dataset, a lot of errors in the second dataset are labeled as “Style/Awkward”. In General, style errors should be harder to detect than simple word mistranslations. Additionally the model seems to make a lot of punctuation errors, where it fails to use punctuation according to German conventions. Aside from these points, some translations marked as wrong lack an obvious error. For example, translating “chancellor” to “Kanzler” cannot be called a translation error.