Institut für Maschinelle Sprachverarbeitung

University of Stuttgart
Pfaffenwaldring 5 b
D–70569 Stuttgart

Masterarbeit

A Visual Analytics Approach for Explainability of Deep Neural Networks

Paul Kuznecov

Course of Study:	Softwaretechnik
Examiner:	Prof. Dr. Ngoc Thang Vu
Supervisor:	Prof. Dr. Ngoc Thang Vu, Dipl.-Inf. Tanja Munz, M.Sc.
Commenced:	April 1, 2018
Completed:	October 1, 2018

Abstract

Deep Learning has advanced the state-of-the-art in many fields, including machine translation, where Neural Machine Translation (NMT) has become the dominant approach in recent years. However, NMT still faces many challenges such as domain adaption, over- and under-translation, and handling long sentences, making the need for human translators apparent. Additionally, NMT systems pose the problems of explainability, interpretability, and interaction with the user, creating a need for better analytics systems. This thesis introduces NMTVis, an integrated Visual Analytics system for NMT aimed at translators. The system supports users in multiple tasks during translation: finding, filtering and selecting machine-generated translations that possibly contain translation errors, interactive post-editing of machine translations, and domain adaption from user corrections to improve the NMT model. Multiple metrics are proposed as a proxy for translation quality to allow users to quickly find sentences for correction using a parallel coordinates plot. Interactive, dynamic graph visualizations are used to enable exploration and post-editing of translation hypotheses by visualizing beam search and attention weights generated by the NMT model. A web-based user study showed that a majority of participants rated the system positively regarding functional effectiveness, ease of interaction and intuitiveness of visualizations. The user study also revealed a preference for NMTVis over traditional text-based translation systems, especially for large documents. Additionally, automated experiments were conducted which showed that using the system can reduce post-editing effort and improve translation quality for domain-specific documents.

Kurzfassung

Deep Learning hat den Stand der Technik in vielen Bereichen, einschließlich der maschinellen Sprachübersetzung, vorangetrieben. In den letzten Jahren ist Neural Machine Translation (NMT) zu dem dominanten Ansatz für maschinelle Sprachübersetzung geworden. Es existiert jedoch noch immer eine Vielzahl von Herausforderungen in NMT, wie beispielsweise Domänenanpassung, Über- und Unterübersetzung, sowie der Umgang mit langen Sätzen. Außerdem haben NMT-Systeme die Probleme der Erklärbarkeit, Interpretierbarkeit und Interaktion mit Endnutzern, was zu einem Bedarf an besseren Analysesysteme führt. In dieser Arbeit wird NMTVis vorgestellt, ein Visual Analytics System für NMT, das an Übersetzer gerichtet ist. Das System unterstützt Nutzer in einer Vielzahl von Aufgaben: dem Finden, Filtern, und Auswählen von fehlerhaften maschinellen Übersetzungen, der interaktiven Nachbearbeitung von Übersetzungen, und der Domänenanpassung des NMT-Modells durch Nutzerkorrekturen. Mehrere Metriken werden eingesetzt, um fehlerhafte Übersetzungen zu detektieren, und mit Parallelen Koordinaten visualisiert. Interaktive, dynamische Graphen-Visualisierungen werden zur Analyse von Übersetzungshypothesen und zur Nachbearbeitung eingesetzt, wobei Beam-Search und Attention-Gewichte des NMT Modells visualisiert werden. Eine web-basierte Nutzerstudie zeigte, dass eine Mehrzahl der Teilnehmer das System positiv in Hinblick auf Effektivität, Benutzbarkeit und Intuitivität der Visualisierungen bewerten. Die Nutzerstudie zeigte zusätzlich eine Präferenz für NMTVis gegenüber traditionellen textbasierten Übersetzungssystemen, insbesondere für große Dokumente. Mehrere automatisierte Experimente belegten außerdem, dass das System zu einer Reduzierung des Arbeitsaufwands in der Nachbearbeitung und Verbesserung der Übersetzungsqualität für domänenspezifische Dokumente führen kann.

Contents

1 Introduction **11**

2 Background **13**
2.1 Machine Learning & Deep Learning . . . 13
2.2 Neural Networks . . . 13
2.3 Recurrent Neural Networks . . . 16
2.4 Long Short-Term Memory . . . 17
2.5 Visual Analytics . . . 19
2.6 Related Works . . . 23

3 Neural Machine Translation **27**
3.1 Introduction . . . 27
3.2 Attention Mechanism & Alignment . . . 28
3.3 Beam Search Decoding . . . 30
3.4 Handling Rare Words . . . 32
3.5 Domain Adaption . . . 34
3.6 Training . . . 34
3.7 Evaluation Metrics . . . 35

4 System Description **37**
4.1 Goals & Requirements . . . 37
4.2 Workflow . . . 40
4.3 Document View . . . 40
4.4 Metrics View . . . 42
4.5 Keyphrase View . . . 46
4.6 Beam Search View . . . 47
4.7 Attention View . . . 53
4.8 Domain Adaption . . . 55
4.9 Implementation . . . 56
4.10 NMT Model . . . 56

5 User Study **63**
5.1 Goals . . . 63
5.2 Study Design . . . 63
5.3 Data Set . . . 64
5.4 Participants . . . 66
5.5 Results . . . 66
5.6 Discussion . . . 69

6 Automated Evaluation **71**
6.1 Data Set 71
6.2 Correlation of Metrics and Translation Quality 72
6.3 Low Translation Quality Experiment 74
6.4 Domain Adaption Experiment 75
6.5 Discussion 79

7 Conclusion **81**

Bibliography **83**

A Questionnaire **89**

List of Figures

2.1 Feedforward Neural Network Structure . . . 14
2.2 Recurrent Neural Network . . . 16
2.3 LSTM Architecture . . . 17
2.4 Visual Analytics Process . . . 19
2.5 Visual Analytics in Deep Learning . . . 20

3.1 Encoder-Decoder Architecture . . . 27
3.2 Attention Mechanism . . . 28
3.3 Alignment Matrix . . . 30
3.4 Beam Search Decoding . . . 31
3.5 BPE Merge Operations . . . 33

4.1 System Overview . . . 39
4.2 Standard System Workflow . . . 40
4.3 Standard System Workflow . . . 41
4.4 Metrics View . . . 42
4.5 Attention Metrics . . . 43
4.6 Metrics View - Usage Scenarios . . . 45
4.7 Keyphrase View . . . 46
4.8 Beam Search View . . . 47
4.9 Beam Search View - Model . . . 48
4.10 Beam Search View - Visual Encoding . . . 49
4.11 Beam Search View - Number of Translations . . . 50
4.12 Beam Search View - Correction . . . 51
4.13 Beam Search View - Custom Correction . . . 51
4.14 Beam Search View - Interaction . . . 52
4.15 Attention View . . . 53
4.16 Attention View - Under-Translation . . . 54
4.17 Attention View - Over-Translation . . . 55
4.18 Domain Adaption . . . 55
4.19 System Architecture . . . 56
4.20 Attentional Encoder-Decoder Architecture . . . 57
4.21 Training and Validation Loss . . . 59
4.22 BLEU Training Scores . . . 60

6.1 Correlation of Metrics and Translation Quality . . . 72
6.2 Distribution of Metric Scores . . . 73
6.3 Low Translation Quality Covered per Metric . . . 74
6.4 BLEU Change in Domain Adaption Experiment . . . 75
6.5 Unigram F_1 Change in Domain Adaption Experiment . . . 76

List of Tables

4.1 Translation Data Sets 58
4.2 NMT Training Results 61

5.1 User Study Document - Sample Sentences 65
5.2 User Study Document - Keyphrases 65
5.3 User Study Results - Participant Background 66
5.4 User Study Results - General Effectiveness 67
5.5 User Study Results - Effectiveness 67
5.6 User Study Results - Visualization 68
5.7 User Study Results - Interaction 68

6.1 Khresmoi Medical Data Set 71
6.2 Translation of Rare Words - Example 78

List of Abbreviations

AP Absentmindedness Penalty

BPE Byte Pair Encoding

CDP Coverage Deviation Penalty

CP Coverage Penalty

EOS End-of-Sequence

GRU Gated Recurrent Unit

LSTM Long Short-Term Memory

ML Machine Learning

NMT Neural Machine Translation

NN Neural Network

RNN Recurrent Neural Network

SOS Start-of-Sequence

VA Visual Analytics

1 Introduction

In recent years, significant progress was made in the field of machine translation [SHB16; VSP+17; WSC+16] through Neural Machine Translation (NMT) due to large data sets, novel techniques, architectures and advances in parallel computing. NMT methods have become state-of-the-art, and are widely employed for translation tasks. However, while significant research was made into improving these models, they still remain "black-boxes", as their internal representation is not suitable for human understanding. For example, NMT models still suffer from issues such as over- and under-translation [TLS+17], where certain words are either translated repeatedly or not translated at all, as well as handling rare words and long sentences [KK17]. These problems show that while NMT systems have achieved promising results in recent years, the need for human translators to correct errors and evaluate machine-generated translations still exists.

Another big challenge NMT systems face is domain adaption [KK17], meaning that NMT models trained on general data sets perform worse at translating domain-specific documents such as medical, legal or scientific texts. However, specific domains are where high-quality machine translation systems are most useful and necessary [CW18]. For example, operational manuals often have thousands of pages, containing specific technical terminology and must often be available in multiple languages, while also being tedious to translate for human translators due to their repetitiveness and large sizes [Hut05]. At the same time, an adequate translation quality is desired without spending too much time or effort. NMT systems have the advantage of being computationally efficient, thus being able to translate large documents quickly but possibly with erroneous translations. Humans, on the other hand, are comparatively slow but can easily spot and correct wrong translations. Therefore, combining NMT and human interaction could help to mitigate these disadvantages of NMT and human translators to get the best of both worlds.

The rapid progress in NMT has created a need for better analytics [KK17], due to the increasing complexity and opaqueness of these systems. In order to successfully combine NMT and human interaction to support users, Visual Analytics (VA) can be used, which is a method for facilitating analytical reasoning by using interactive interfaces. VA can be employed to support users for finding erroneous translations, collaborative translation, and post-editing. Existing approaches applying visualizations to NMT have focused on aiding researchers for debugging or gaining insight into models [SGPR18] [SGB+18], or only focused on a small part of the translation process such as visualizing beam search [LSK17] or attention weights [RFB17]. However, integrated approaches that tackle multiple challenges in translating large documents, i.e. pre-translation, sentence selection and interactive post-editing targeting end users are needed. NMT also offers new opportunities for interaction and visualization through techniques such as the attention mechanism and beam search decoding.

This thesis aims at developing an integrated VA approach for translating domain-specific documents using an NMT model. The proposed system should help users to find erroneous translations, and aid in post-editing using interactive visualizations. The system should adapt to a domain by incorporating user corrections and iteratively improve itself. Finally, the system should keep the human user in the loop, making the underlying NMT model interpretable and explainable.

The main contributions of this thesis are:

1. Proposal of a novel VA approach for interactive post-editing with NMT models
2. Development of a prototype using an attentional encoder-decoder NMT model implementing the proposed approach
3. Development of interactive visualizations and techniques for exploring documents, translation hypotheses, and NMT model aspects including beam search and attention weights
4. Evaluation through a user study and automated experiments to validate the proposed approach

Structure

This thesis is structured as follows:

Chapter 2 – Background gives an overview over relevant background topics related to Machine Learning, different types of Neural Networks, Visual Analytics and related works.

Chapter 3 – Neural Machine Translation gives a brief introduction to Neural Machine Translation, as well as related techniques and challenges that are relevant for the proposed system, such as the attention mechanism, domain adaption and handling rare words.

Chapter 4 – System Description introduces the proposed system called NMTVis and describes its goals, design, architecture and implementation.

Chapter 5 – User Study describes the web-based user study and discusses the results.

Chapter 6 – Automated Evaluation describes the automated experiments that were conducted concerning the effectiveness of the system.

Chapter 7 – Conclusion summarizes the findings of this thesis and discusses future research directions.

2 Background

This chapter will give additional background for relevant topics and introduce the most important theoretical foundations related to Machine Learning that are needed to understand Neural Machine Translation. First, Machine Learning and Deep Learning are introduced, and Neural Networks are explained, including forward- and backpropagation. Then, Recurrent Neural Networks are described, followed by a short introduction to the Long Short-Term Memory architecture. A brief introduction to Visual Analytics and approaches for Deep Learning is given. Finally, related works are summarized and discussed in the context of this thesis.

2.1 Machine Learning & Deep Learning

Machine Learning (ML) is a research field that concerns itself with algorithms and techniques to extract patterns from raw data [GBC16], in order to create *models* that can perform tasks without being explicitly programmed for them. This enables a new computational paradigm, where computer programs can perform cognitive tasks such as speech recognition, translation, or image labeling by learning from experience. Computers learn a hierarchy of concepts, where each concept is built from simpler concepts. Concepts can thus be seen as layered on top of each other, starting from simple concepts and building up to complex concepts. For example, in machine translation, a model may first learn the semantic meaning of each word, and then how words are related within a sentence based on their meaning, and then how words from different languages relate to each other. Deep Learning is a subfield of ML that represents the world as a nested hierarchy of concepts, where more abstract representations are computed from less abstract ones [GBC16]. Neural Networks are at the core of Deep Learning, as they directly model these deep hierarchies of concepts using multiple layers.

2.2 Neural Networks

A Neural Network (NN) is a type of non-linear model consisting of artificial neurons [Sch97]. Due to their ability to approximate any function on their input to any desired accuracy, they are regarded as ”universal approximators” [HSW89], making them useful for a wide variety of tasks. They can be used for classification, i.e. classifying data into predetermined classes, or regression, i.e. predicting some real-valued output. Neurons, which are vaguely inspired by biological neurons, are the basic building block of NNs. Feedforward NNs are the simplest type of NN, taking some input and computing output. This computation is defined by the structure of the NN, as well as the choice of so-called *activation functions*, which are applied during computation.

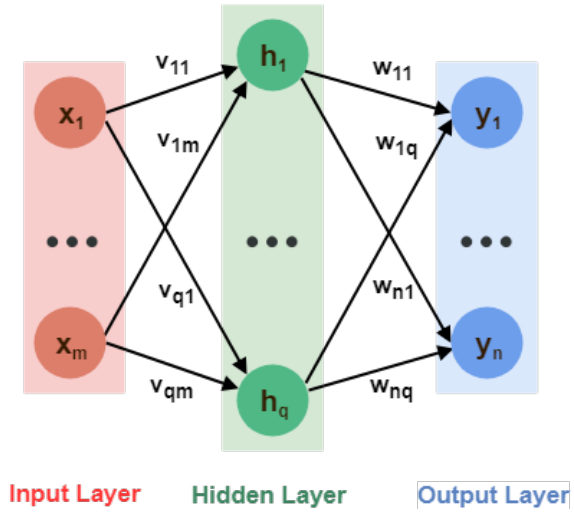

Figure 2.1: The structure of a feedforward NN with single hidden layer. Intermediate nodes within layers are left out for visual clarity.

Figure 2.1 shows the typical structure of a feedforward NN. The structure of an NN can be viewed as a directed, weighted graph, with neurons acting as vertices and connections between neurons as edges with specific weights. An NN consists of multiple *layers*, which are related sets of neurons at a certain depth in the NN. No connections are allowed between nodes within a layer, and nodes can only be connected to nodes in subsequent layers. The input data is fed into the *input layer*, which simply stores the input. Then, at least one *hidden layer* follows, which can be seen as an abstract representation of the input data that is learned by the NN. Multiple hidden layers can be stacked, each with possibly distinct sizes. Finally, the output layer contains the output of the NN, computed from the last hidden layer.

2.2.1 Forward Propagation

We now want to focus on how exactly the input is transformed into the output in a feedforward NN, a process also called *forward propagation*. Let us assume that the input of the NN is a vector $\boldsymbol{x}$ of size m, we want to transform it into the output vector $\boldsymbol{y}$ of size n. For simplicity, let us assume we have a single hidden layer of size q, with activation function f, and output activation function g. In practice, the activation function is often a sigmoid, hyperbolic tangent, or another non-linear function. The value h_j of the j-th node in the hidden layer is computed as follows:

$$h_j = f(\sum_{i=1}^{m} v_{ji} x_i) \tag{2.1}$$

where v_{ji} is the weight between the i-th input node and j-th hidden node. We see that h_j is computed by applying the activation function on the weighted sum of the input values x_i. We can also consider the weights between two layers as a single matrix, which simplifies the computation to simple matrix multiplication. Let $\boldsymbol{W_{hx}}$ be a weight matrix of size $q \times m$ between the input layer and the hidden layer, with element w_{ji} being the weight between the i-th input node and j-th hidden layer node. Then the values of the hidden layer can be expressed as a vector $\boldsymbol{h}$ as follows:

$$\boldsymbol{h} = f(\boldsymbol{W_{hx}}\boldsymbol{x}) \tag{2.2}$$

Note that the activation function must now be applied element-wise to the vector resulting from the matrix multiplication. As we can see, the value of a hidden node is computed as the weighted sum of each node from the input layer, with the activation function applied to this sum. As such, these weights represent how much a certain input feature x_i should influence the value of a hidden node. Similarly, the value y_j of the j-th node in the output layer can be computed as such:

$$y_j = g(\sum_{i=1}^{q} w_{ji} h_i) \tag{2.3}$$

Again, we can vectorize the previous equation to compute the output vector $\boldsymbol{y}$ directly:

$$\boldsymbol{y} = g(\boldsymbol{W_{yh}}\boldsymbol{h}) \tag{2.4}$$

Here, $\boldsymbol{W_{yh}}$ is a weight matrix of size $n \times q$ representing the weights between the hidden layer and the output layer. As before, we take weighted sums of the values from the previous layer, apply an activation function, and receive the values for the output layer.

2.2.2 Backpropagation

Forward propagation is the process of computing the output of an NN given some input and weights. The question still remains how to determine weights such that the NN computes some desired output for a given task. *Backpropagation* [RHW86] is a method for adjusting the weights of an NN, based on repeated application of the chain rule for derivatives. Errors from the output layer are propagated *backwards* through the network, layer by layer, resulting in the weights to be updated to minimize the error. Formally, we define a loss (or error) function L which measures how far off the output of the NN $\hat{\boldsymbol{y}}$ is given training sample $(\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{x}$ is the input and $\boldsymbol{y}$ is the true output or ground truth. A loss function that is often used for NMT is the negative log-likelihood:

$$L(\hat{y_j}, y_j) = -\hat{y_j} \log(y_j) \tag{2.5}$$

The loss tends towards negative infinity when $\hat{y_j}$ is 1 and y_j is close to 0. On the other hand, the loss is 0 whenever $\hat{y_j}$ is 0, regardless of the value of y_j. If we take the ground truth and NN output as probabilities, the negative log-likelihood can be seen as a penalty for assigning low probabilities for outputs that should have high probability. Let $r_j = \sum_{i=1}^{q} w_{ji} h_i$ be the weighted sum of all hidden node values, i.e. the value of the j-th output node before applying the activation function. The effect of a weight w_{ji} on the loss can then be expressed as the partial derivative $\frac{\delta L}{\delta w_{ji}}$, which can be computed as follows using the chain rule:

$$\frac{\delta L}{\delta w_{ji}} = \frac{\delta L}{\delta y_j} \frac{\delta y_j}{\delta r_j} \frac{\delta r_j}{\delta w_{ji}} \tag{2.6}$$

The goal of training is to minimize the loss function L so that the output of the NN is as close to the true output as possible. Because L is a function of the NN weights $\boldsymbol{w}$, we can differentiate it with respect to each weight. Because the derivative can usually not be given analytically, an iterative approach called *gradient descent* is used to find the minimum of the loss function. Given a starting point, i.e. initial values for the weights, the gradient of the loss function is repeatedly computed with respect to the weights. Because the gradient points in the direction of the steepest ascent, the vector of same magnitude and opposite direction is added to iteratively refine the estimate for each weight w_i of the NN:

$$\Delta w_i = -\eta \frac{\delta L}{\delta w_i} \tag{2.7}$$

$$w_i^{t+1} = w_i^t + \Delta w_i \tag{2.8}$$

where w_i^t is the weight at time step t, and w_i^{t+1} at time step $t + 1$ after a single update operation of gradient descent. The learning rate η controls how much each update step in the gradient descent changes the weights and is an important hyperparameter for training NNs. A large learning rate may overshoot the actual minimum by taking too large steps, while a small learning rate may get trapped in a local minimum or take a long time to converge to a solution.

2.3 Recurrent Neural Networks

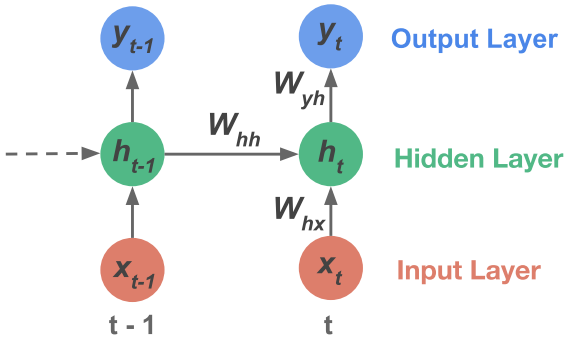

Figure 2.2: An unrolled RNN showing how output at time t is computed based on the previous timestep.

Feedforward NN models typically cannot capture temporal information, meaning that separate predictions do not influence each other and their order does not matter. For tasks involving predicting a variable-length output sequence given a variable-length input sequence, these feedforward NNs are not suitable, as they do not take previous predictions into account. In translation tasks, for example, a partial translation of previously predicted words is necessary to provide context when predicting the next word. Recurrent Neural Networks (RNNs) are an extension of feedforward NNs where the hidden state of the previous timestep is passed as additional input to the network. By keeping the hidden state and passing it to the next timestep, the model can take previous inputs and predictions into account. The forward pass of an RNN can be explained by the following two equations (based on [Lip15]):

$$\boldsymbol{h_t} = f(\boldsymbol{W_{hx}}\boldsymbol{x_t} + \boldsymbol{W_{hh}}\boldsymbol{h_{t-1}} + \boldsymbol{b_h}) \tag{2.9}$$

$$\boldsymbol{y_t} = g(\boldsymbol{W_{yh}}\boldsymbol{h_t} + \boldsymbol{b_y}) \tag{2.10}$$

where $\boldsymbol{W_{hx}}$ is the weight matrix that connects the input to the hidden state and $\boldsymbol{W_{hh}}$ is the weight matrix that connects the hidden state of subsequent timesteps and f is an arbitrary activation function, and $\boldsymbol{b_h}$ and $\boldsymbol{b_y}$ are biases. Equation 2.9 explains how the hidden state $\boldsymbol{h_t}$ is computed in a RNN by adding the two terms $\boldsymbol{W_{hx}}\boldsymbol{x_t}$ and $\boldsymbol{W_{hh}}\boldsymbol{h_{t-1}}$. Note how this is an extension of a feedforward NN by adding connections between the nodes of the hidden layers of subsequent timesteps, as seen in Figure 2.2. These connections between timesteps make RNNs recurrent, allowing them to learn long-term dependencies. The output $\boldsymbol{y_t}$ is calculated by multiplying the weight matrix $\boldsymbol{W_{yh}}$ and the hidden state of the current timestep $\boldsymbol{h_t}$ and applying a function g. It is also important to realize that all weight matrices are *shared* between all timesteps, which is critical for training RNNs and allows to use backpropagation for training.

2.4 Long Short-Term Memory

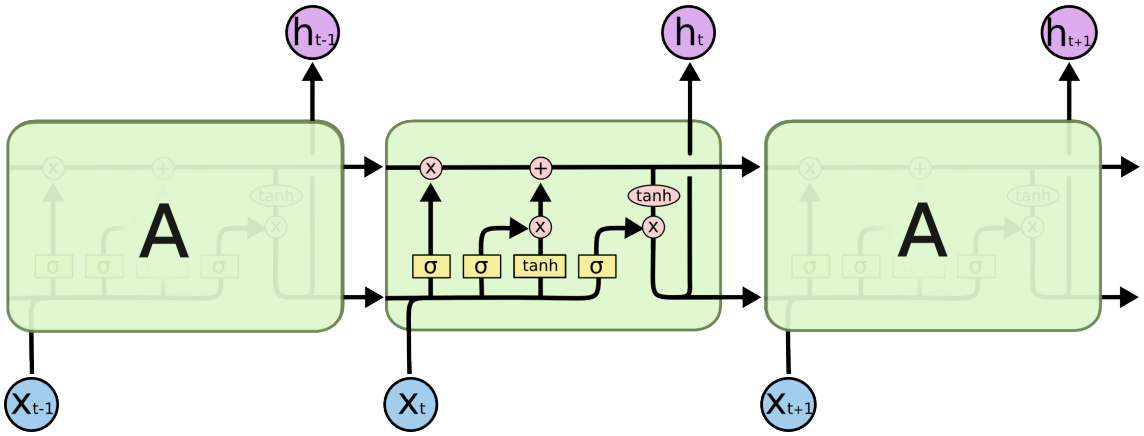

Figure 2.3: A standard LSTM architecture showing the gates, layers, operations and the data flow inside an LSTM cell [Ola15].

Long Short-Term Memory (LSTM) [HS97] is an RNN architecture that solves the two largest problems of ordinary RNNs, namely (1) exploding gradients and (2) vanishing gradients. These problems hinder RNNs from learning long-term dependencies in data, which is especially relevant in NMT, as languages naturally have such dependencies. The main mechanisms by which LSTMs solve these problems are *gates*, which control the flow of information and *cell states* which store information about long-term dependencies. The LSTM *cell* is the basic building block of a LSTM, analogous to a neural unit inside an RNN. Figure 2.3 shows a cell and its inner structure. Each cell takes the previous hidden state $\boldsymbol{h_{t-1}}$, cell state $\boldsymbol{C_{t-1}}$ and current source word $\boldsymbol{x_t}$ as inputs, and produces a new hidden state $\boldsymbol{h_t}$ and cell state $\boldsymbol{C_t}$.

We will now explain how an LSTM cell computes new outputs based on its previous state. First, the forget gate decides which parts and how much of the previous cell state should be forgotten. For this reason, a sigmoid layer is used, which outputs values between 0 and 1, that is later multiplied with the cell state element-wise. High values indicate that a certain part of the cell state should remain, while low values indicate that it should be forgotten. These values are computed by concatenating the previous hidden state $\boldsymbol{h_{t-1}}$ and the source token $\boldsymbol{x_t}$, multiplying by a weight matrix $\boldsymbol{W_f}$, adding a bias $\boldsymbol{b_f}$ and finally passing the result through the sigmoid function:

$$\boldsymbol{f_t} = \sigma(\boldsymbol{W_f} \cdot [\boldsymbol{h_{t-1}}, \boldsymbol{x_t}] + \boldsymbol{b_f}) \tag{2.11}$$

Then the input gate decides which incoming information to store in the cell state. As before, these values are computed by a sigmoid layer using the weight matrix $\boldsymbol{W_i}$ and bias $\boldsymbol{b_i}$:

$$\boldsymbol{i_t} = \sigma(\boldsymbol{W_i} \cdot [\boldsymbol{h_{t-1}}, \boldsymbol{x_t}] + \boldsymbol{b_i}) \tag{2.12}$$

High values in i_t mean that this part of the input will be updated more than parts with low values. Having computed which parts of the cell state have to be updated, we can compute $\tilde{\boldsymbol{C}}_{\boldsymbol{t}}$ which contains the actual update information:

$$\tilde{\boldsymbol{C}}_{\boldsymbol{t}} = \tanh(\boldsymbol{W_C} \cdot [\boldsymbol{h_{t-1}}, \boldsymbol{x_t}] + \boldsymbol{b_C}) \tag{2.13}$$

Having decided which parts of the cell state to forget and to update, we can apply these changes and compute the new cell state $\boldsymbol{C_t}$ by combining what to forget ($\boldsymbol{f_t} \odot \boldsymbol{C_{t-1}}$) and what to update ($\boldsymbol{i_t} \odot \tilde{\boldsymbol{C}}_{\boldsymbol{t}}$):

$$\boldsymbol{C_t} = \boldsymbol{f_t} \odot \boldsymbol{C_{t-1}} + \boldsymbol{i_t} \odot \tilde{\boldsymbol{C}}_{\boldsymbol{t}} \tag{2.14}$$

Next, the actual output of the cell $\boldsymbol{h_t}$ needs to be computed based on the current cell state $\boldsymbol{C_t}$. Once again, we pass the concatenation of $\boldsymbol{x_t}$ and $\boldsymbol{h_{t-1}}$ through a sigmoid layer to compute $\boldsymbol{o_t}$:

$$\boldsymbol{o_t} = \sigma(\boldsymbol{W_o} \cdot [\boldsymbol{h_{t-1}}, \boldsymbol{x_t}] + \boldsymbol{b_o}) \tag{2.15}$$

This is the output gate and decides how much each part of the cell state will be used as output. Finally, $\boldsymbol{h_t}$ is computed by element-wise multiplication of $\boldsymbol{o_t}$ and $\boldsymbol{C_t}$:

$$\boldsymbol{h_t} = \boldsymbol{o_t} \odot \boldsymbol{C_t} \tag{2.16}$$

Together, forget, input and output gates make it possible for LSTMs to learn long-term dependencies by allowing the NN to decide what to forget, update and output. Many variants of LSTMs exist, changing the connections of gates or layers inside the cells. The most notable variation is the Gated Recurrent Unit (GRU) [CVG+14], which can be seen as a simplified type of LSTM, and which has shown great success in sequence modeling tasks [CGCB14]. A GRU has only two gates, a reset gate and an update gate, and consolidates cell state and hidden state. Its main advantages over LSTM are simpler computation and implementation.

2.5 Visual Analytics

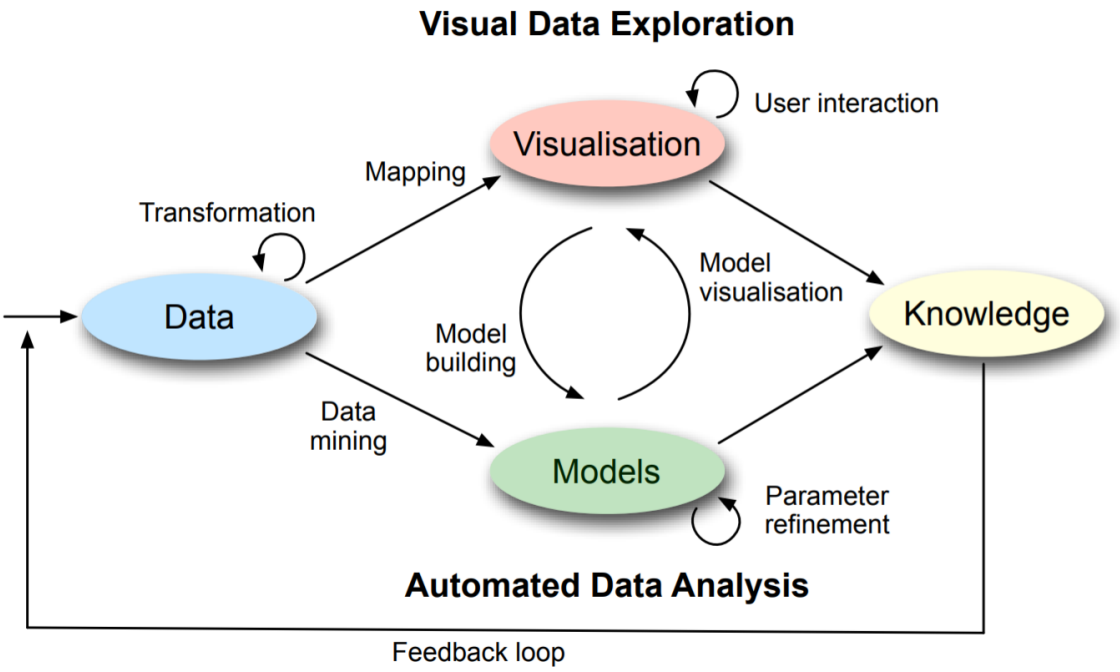

Figure 2.4: The Visual Analytics process [KKE10]. Models and visualisations aid in knowledge generation and form a feedback loop back to the data.

This section introduces Visual Analytics in general, as well as its application in the Deep Learning domain. Different approaches and techniques for Visual Analytics in Deep Learning will be explained and put into the context of this thesis.

2.5.1 Definition

Visual Analytics (VA) is "the science of analytical reasoning facilitated by interactive visual interfaces" [CT05]. Keim et al. [KAF+08] further define the goals of VA as the creation of tools and techniques to enable people to:

1. Synthesize information and derive insight from massive, dynamic, ambiguous, and often conflicting data
2. Detect the expected and discover the unexpected
3. Provide timely, defensible, and understandable assessments
4. Communicate assessment effectively for action

The VA process [KKE10] is at the core of any VA system. Figure 2.4 shows how the concepts of *data*, *models*, *visualisation* and *knowledge* are related. The process starts with some data that a user wants to analyze, which first be transformed, i.e. aggregated or filtered in some way. This data is mapped to different visualisations allowing the user to analyze different facets or relationships in

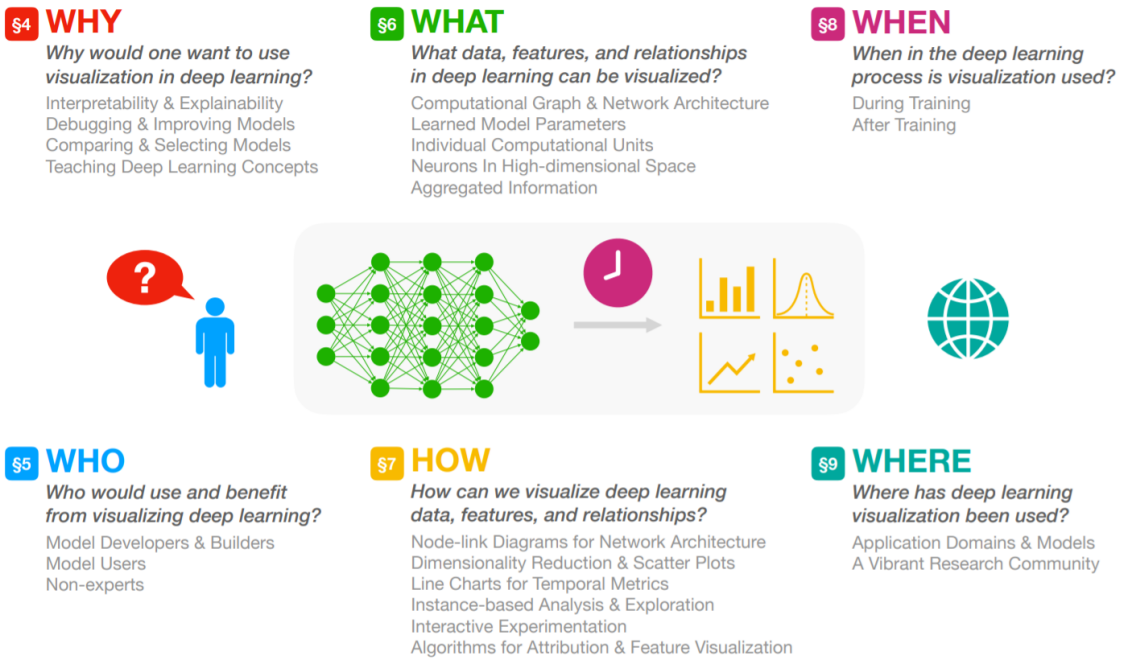

Figure 2.5: Conceptual framework of Hohman et al. [HKPC18] for VA in Deep Learning. Techniques, approaches and systems can be categorized by answering the questions *why*, *what*, *when*, *who*, *how* and *where* to use visualizations in Deep Learning.

the data. Models can be built from this data which can also be visualized in addition to the data. User interaction can update this model, leading to an update in the visualization. While interacting with the model through the visualization, the user hopefully gains some knowledge about the task he wants to achieve, forming a feedback loop back into the data, e.g. by choosing different data or changing it in some way.

We will now explain how the VA process is applied in this thesis. The data is a large domain-specific document that the user wants to translate into a different language. Sentences are preprocessed, filtered and aggregated to allow translation and find erroneous translations. Sentences and translation hypotheses are mapped to multiple visualisations. The model in the VA process is the NMT model trained from large text corpora and used to translate sentences. Model visualisation is done by showing the beam search tree, allowing the user to directly interact with the model. Parameters of the model such as beam size can be refined by the user, which reflect through changes in the visualisations. Finally, the user gains some knowledge by analyzing different translation hypotheses and correcting machine-generated translation, feeding corrections back to the NMT model, which in turn updates translations of the original document. This forms a feedback loop between knowledge and data, which is a central part of the VA process.

2.5.2 Visual Analytics in Deep Learning

Deep Learning comes with a special set of problems and challenges for applying VA. Hohman et al. [HKPC18] conducted a comprehensive survey on VA in Deep Learning and introduced a conceptual framework for analyzing and categorizing techniques and approaches as seen in Figure 2.5. They pose six questions (why, who, what, how, when and where) that can be used to describe VA approaches. The following four reasons for using visualization in Deep Learning are identified:

Interpretability & Explainability Internal model representations can be visualized to make the model predictions interpretable for humans, allowing users to see reasons why a model came to a certain prediction.

Debugging & Improving Models Systematic error cases can be found through visualizations in order to improve models.

Comparing & Selecting Models Visualizations can be used to compare models regarding their performance to aid in model selection.

Teaching Deep Learning Concepts Visualizations can be useful to teach Deep Learning concepts such as NN architectures and computational graphs.

This thesis focuses on interpretability and explainability as the main reason for using VA, as NMT has major potential for improving the translation workflow. Improving models is another important focus of our system, due to the possibility to use manual corrections made by users. However, we focus on improving the model by updating model weights, rather than improvement through architectures or algorithms.

Additionally, there are two temporal aspects for applying VA in Deep Learning:

During Training Visualizations can be used during training to show statistics about a model over time such as accuracy, loss or evaluation metrics. This may be useful to compare models during training or find out if a model is overfitting and training should be stopped.

After Training Visualizations can be used to visualize models after training, e.g. for visualizing model architecture or computational graphs.

The proposed system is not relevant during initial training of the NMT model. However, the system adapts to a domain by incorporating corrections made by the user in an online fashion. As such, our approach combines both temporal aspects and therefore is relevant during and after training.

An important aspect for VA systems is user context, which explains who the users are, what their goals are and what environment they work in. In the context of Deep Learning, users can be classified into one of the following groups:

Model Developers & Builders These are domain experts in Deep Learning, often researchers and engineers, who create, implement and deploy new Deep Learning architectures and techniques. Their focus for using visualizations is technical in nature. They might want to visualize performance metrics across time to compare models or visualize complex NN architectures.

Model Users This group includes technical people who work with Deep Learning models but do not develop new models or techniques. For example, software engineers who integrate a trained model for a domain-specific application are part of this group. Their focus is more on understanding and interpreting model behavior than improving models.

Non-experts Non-experts are people who do not possess special knowledge about Deep Learning that may use a system based on it. Visualization for non-experts can be used to make predictions made by Deep Learning systems explainable or for educational reasons, such as for teaching Deep Learning concepts.

This thesis is primarily concerned with non-experts for the proposed VA system. As such, we are targeting translators, rather than researchers or developers of NMT model. While there are already tools for model developers and users to analyze NMT models, end users, i.e. translators, may also benefit from visualizing model behavior.

Another important issue is what aspects of a Deep Learning model should be visualized:

Computational Graph & Network Architecture The computational graph, which defines how data moves throughout an NN, can be visualized to give model developers an overview of computations and their relation for a given model. Additionally, the NN architecture, including layers and their connections can be visualized.

Learned Model Parameters In contrast to the previous topic of visualizing NN structure, the learned model parameters, i.e. weights, can be also be visualized. As these weights are learned from data, visualizing them may show what a particular NN model has learned.

Individual Computation Units We can also visualize aspects of the individual computation units of NNs, i.e. single neurons. Activations of neurons can be visualized for a given instance to show features representations generated by the model. Additionally, gradients can be computed during backpropagation, and visualizing these for neurons could be useful to see how errors are propagated throughout the NN.

Neurons in High-dimensional Space NNs compute high-level representations of input data through multiple hidden layers. These representations can be seen as high-dimensional vectors, that in turn can be visualized e.g. by projecting them onto a 2D plane. This could, for example, help to see if an NN model learns useful word embeddings in NMT by checking if semantically related words are close together.

Aggregated Information Finally, aggregated information of a Deep Learning model can be visualized. This includes aggregation of activations for multiple instances or evaluation metrics that measure the performance of a model regarding some task.

Because the system proposed in this thesis targets non-experts, we chose not to visualize NN architecture or learned model parameters. Instead, aggregated information is visualized that correlate with translation quality.

Finally, having clarified what can be visualized for whom and for what reasons in Deep Learning, the following types of approaches can be used to answer the question of how to visualize Deep Learning:

Node-link Diagrams for Network Architecture NNs consist of neurons and weights that connect nodes. Therefore, they can be visualized as node-link diagrams by representing neurons as nodes and weights as directed, weighted links. The magnitude of a weight can be mapped to line color or thickness. For computational graphs, operations are represented as nodes and links represent the flow of data.

Dimensionality Reduction & Scatter Plots As mentioned for the *Aggregated Information* aspect for what to visualize, representations that are learned by NN models are useful to see what a mode has learned. As these representations are high-dimensional, dimensionality reduction can be used to map representations onto a lower-dimensional space. Then, scatter plots can be used to show these low-dimensional representations for different instances.

Line Charts for Temporal Metrics Line charts are a simple visualization method to show how model metrics such as accuracy or loss change over time. This can be helpful during model training, e.g. to see if a model starts to overfit when the validation error stops decreasing. Also, this technique enables model comparison and selection by visualizing metrics for multiple models at once.

Instance-based Analysis & Exploration Instance-based Analysis focuses on small-scale model exploration through specific data instances. Here, the model is observed during prediction for a single instance, which can be more tractable compared to analyzing model performance on multiple instances at once. Testing is another important use case for instance-based analysis. For instances where the model does not produce a correct classification or prediction, it is important to analyze how or why the model failed to produce the correct output.

Interactive Experimentation Interactive exploration enables users to change aspects of a model to see what changes this causes. One way is by allowing users to input arbitrary data to the model and visualize how the model responds. Another method enables users to change model hyperparameters such as learning rate or dropout, and directly see how this affects results.

Algorithms for Attribution & Feature Visualization Finally, algorithms for attribution and feature visualizations show what parts of the input were important for a certain classification or prediction made by an NN model. Feature visualization constructs synthetic input that is supposed to represent a certain class best. Other techniques use heatmaps overlaid over original input to highlight important regions of the input, which enables visualization of attribution.

This thesis applies VA mainly through instance-based analysis, interactive experimentation, and algorithms for attribution. Incorrect translations are the instances that are analyzed and interactively explored by users, using dynamic graph visualizations afforded by beam search decoding. We also visualize attention weights to show how source words are attributed to each translated word.

2.6 Related Works

The following section will introduce related works on the topics of NMT and VA, as well as how they relate to this thesis. Related works can be grouped into two distinct groups: (1) algorithms, architectures, and approaches for machine translation or post-editing, that try to improve the

translation quality or extend the capability of models and systems in some way, and (2) interactive visualization and VA approaches relating to NNs and NMT, which try to give insight into models and their behavior.

2.6.1 Machine Translation

Knowles and Koehn [KK16] developed an interactive translation prediction method for NMT models. Instead of post-editing machine-generated translations, users type in translations from scratch and the system suggests words after each keystroke. Users can then either select the suggested word or proceed manually typing. As in this thesis, they use an attentional encoder-decoder model, BPE and beam search decoding, also focusing on German→English translation. The goals of their system are strongly aligned with this thesis, mainly supporting translators in an online fashion and allow interactive translation using an NMT model. They conducted an automated evaluation, showing significant improvement regarding word and letter prediction accuracy of their method compared to traditional phrase-based systems, proving the feasibility of an interactive NMT system. However, they point out that computational efficiency is still a major drawback of NMT based system due to their large computational overhead. Their results can be seen as a form of theoretical validation for the implementation of an interactive NMT system.

Peris et al. [PCC17] proposed and evaluated multiple optimization techniques for online learning of NMT models in a post-editing scenario. They show that adaptive Stochastic Gradient Descent methods such as Adam [KB14] are well suited for online domain adaption of NMT models. Their work is highly relevant for this thesis, as adapting an NMT model based on users corrections is valuable in a VA setting. We used their findings to guide the development of our domain adaption and fine-tuning approach.

Hokamp and Liu [HL17] introduced grid beam search, which extends grid search by lexical constraints. During decoding, the algorithm forces beam search to generate the lexical constraints, guaranteeing that they appear in the output sentence. These constraints might come from user input during interactive post-editing of translations. They show that grid beam search can improve translation quality in interactive scenarios and for domain adaption. A simplified version of this algorithm was used in this work, using only prefix constraints to force decode a partial translation for beam search decoding.

Cheng et al. [CHC+16] introduced the pick-revise framework for interactive post-editing. Users first pick an incorrectly translated phrase and then revise it with a correct translation, after which the model re-translates the original sentences taking into account the revisions as constraints. This work can be seen as an extension of their work by using VA to improve the post-editing task using a graph visualization to guide the user.

2.6.2 Interactive Visualizations & Visual Analytics

Recently, combining Deep Learning and VA has become an emerging research topic. Strobelt et al. [SGPR18] have developed LSTMVis, an interactive, web-based tool for analyzing LSTMs. LSTMVis visualizes the hidden states dynamics and enables users to formulate hypotheses about the model. Several use cases such as machine translation, phrase separation, and biological sequence

analysis are provided to show the utility of the system. While LSTMVis aims to provide researchers and machine learning developers with a way to debug LSTM models, this thesis targets end users for assisting them during post-editing of machine-translated documents.

Seq2Seq-Vis [SGB+18] is another tool for interactively debugging sequence-to-sequence models for textual problems such as translation and part-of-speech tagging. Its main goal is giving researchers insights into the model's behavior and internal state on multiple levels such as encoding, decoding, attention, and prediction to find out on which level a model behaves erroneously. Their work also explores interactive beam search and modification of attention weights by the user However, it does not allow substitution of arbitrary words, phrases or even tokens unknown to the model. Unlike the system proposed in this thesis, Seq2Seq-Vis is completely sentence-based: users can input single sentences and inspect model behavior, but it does not give any document-level insights about a model. Finally, Seq2Seq-Vis does not persist users' modifications of attention or beam search decoding, and does not feed corrections back into the model for improvement or domain adaption.

Lee et al. [LSK17] introduced a system for interactive beam search decoding, supporting both exploration of beam search decoding and manual and automatic adjustment of attention weights. We use the main idea of visualizing translation hypotheses from beam search decoding as a tree in this thesis. However, their system uses only basic visual encoding and interaction techniques, while this work explores interactive beam search visualization in depth. Their visualization also does not handle subword units transparently, instead of showing raw subword units in the beam search tree. Beam search is used exclusively for hypothesis exploration in their work, without allowing post-editing or translation constraints generated by the user. They also do not deal with the replacement of unknown words, and only focus on single sentences, as opposed to the document-based approach in this work. Additionally, they did not conduct any evaluation or validation of their approach.

Rikters et al. [RFB17] developed a web-based tool for visualizing attention weights and introduced multiple metrics for automatically scoring sentences based on attention without any reference translations. Importantly, they suggested multiple attention-based metrics that are supposed to correlate with translation quality and implemented a prototypical visualization system allowing users to sort sentences based on metrics. Unfortunately, no type of evaluation or validation was described in their approach, therefore a statistical evaluation of these metrics was conducted as part of this thesis. We use the proposed attention-based metrics as the basis for sentence selection, to allow users to filter sentences that are likely to contain translation errors without having access to reference translations. While Rikters et al. [RFB17] use multiple bar charts to visualize metric scores, the proposed system uses a parallel coordinates plot that is better suited to visualize multidimensional data and allows easier filtering and interaction.

3 Neural Machine Translation

This chapter will give a brief introduction into Neural Machine Translation and important topics including evaluation metrics, handling rare words, attention mechanism and beam search decoding. The relation of these topics and their relevance to the proposed system will also be described.

3.1 Introduction

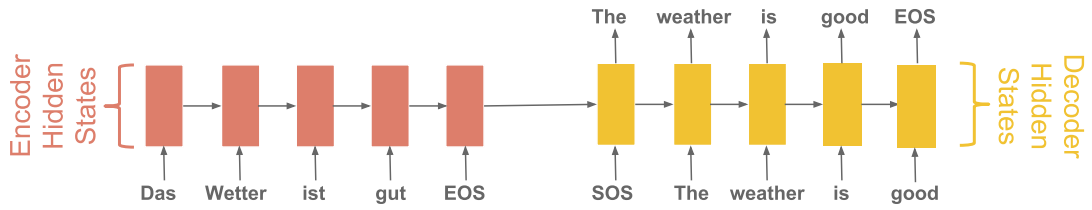

Figure 3.1: A sample translation by the RNN encoder-decoder architecture as proposed by Cho et al. [CMG+14]. Note that the last hidden state of the encoder (red) is used to initialize the hidden state of the decoder (yellow).

Neural Machine Translation (NMT) is a sub-field of machine translation that concerns itself with techniques using NNs. In general, the goal of machine translation is to translate a source sequence $\boldsymbol{x} = \{x_1, \ldots, x_M\}$ into a target sequence $\boldsymbol{y} = \{y_1, \ldots, y_N\}$ where each x_i is a source token from a source vocabulary V_s, and each y_i is a target token from a target vocabulary V_t. We are seeking a model parametrized by weights $\boldsymbol{\theta}$ that maximizes the conditional probability $p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})$. Because the target translation is generated one token at a time, this probability can be decomposed as follows [LPM15]:

$$p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}) = \prod_{j=1}^{N} p_{\boldsymbol{\theta}}(y_j|y_1 \ldots y_{j-1}, \boldsymbol{x}) \tag{3.1}$$

This decomposition implies that the probability of each translated token depends only on the previously translated words and the source sequence. In order to take previous translations into account when predicting the next word, the model must learn long-term dependencies of the data. As explained in Chapter 2, RNNs and LSTMs in particular are suitable for tasks where such dependencies have to be learned. Therefore, many recent approaches in NMT are based on RNN architectures. One of the first NN architectures for NMT was introduced by Cho et al. [CMG+14]. This architecture is made up of two RNNs, an *encoder*, and a *decoder*. The encoder is responsible for encoding the source sequence into a fixed-length vector, while the decoder generates a target sequence after reading this vector.

Figure 3.1 shows how the encoder reads the source sequence, passing the previous hidden state to the next timestep and finally creating a last hidden state, which is then fed into the decoder along with the Start-of-Sequence (SOS) token. The decoder then generates target tokens until the End-of-Sequence (EOS) token is generated, stopping the translation process. At every timestep, the previously generated token and the previous hidden state is passed to the decoder. The major downside to this basic encoder-decoder architecture is the fact that the entire source sequence must be encoded into a single fixed-length vector. This creates a bottleneck for translating long sequences and limits the effectiveness of the model.

3.2 Attention Mechanism & Alignment

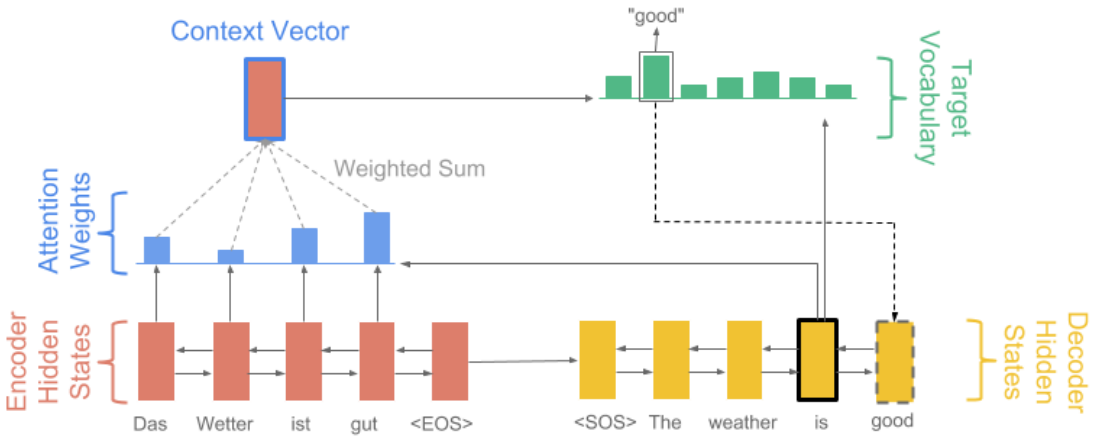

Figure 3.2: The attention mechanism as used in a sequence-to-sequence model. The current decoder hidden state from the target word *is* is used along with the encoder hidden states to compute the context vector and the next predicted word *good*. The figure was adapted from See et al. [SLM17].

Bahdanau et al. [BCB14] first introduced the attention mechanism for NMT. In the traditional encoder-decoder approach, the encoder must encode all relevant information into a single fixed-size vector, as the decoder only has access to the last hidden state of the encoder. The attention mechanism allows sequence-to-sequence models to *attend* to different parts of the source sequence while predicting the next element of the target sequence, by giving the decoder access to the hidden states of the encoder.

Figure 3.2 shows how the NMT model computes and uses attention to predict the next token. First, the encoder consumes the complete input sequence, producing one hidden state per token. In this architecture, the encoder is *bidirectional*, which means that an encoder hidden state $\boldsymbol{h_s}$ is produced by concatenating the forward hidden state $\overrightarrow{\boldsymbol{h_s}}$ and backward hidden state $\overleftarrow{\boldsymbol{h_s}}$, which are produced by a forward RNN reading the source sequence from left to right, and a backward RNN reading it from right to left, respectively. During decoding, the current decoder hidden state and the encoder hidden

states are used to compute the attention weights, which are then used to compute the context vector as a weighted sum of the encoder hidden states. Finally, the context vector and current decoder state are used to produce the output layer, giving a probability distribution over the target vocabulary.

Formally, let $(\boldsymbol{h_1}, \ldots, \boldsymbol{h_N})$ be the hidden states of the encoder after reading each of the N source elements. Then, for each pair of source and target elements $(\boldsymbol{x_j}, \boldsymbol{y_i})$ a weight α_{ij} is computed as follows, with a being the *alignment* model:

$$e_{ij} = a(\boldsymbol{s_{i-1}}, \boldsymbol{h_j}) \tag{3.2}$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{N} \exp(e^{ik})} \tag{3.3}$$

This alignment model, which is parametrized as a feedforward NN, expresses how well each x_j and y_i match, and depends on the previous decoder hidden state $\boldsymbol{s_{i-1}}$ and encoder hidden state $\boldsymbol{h_j}$. Using this alignment, we can finally compute the *context vector* $\boldsymbol{c_i}$, which is a weighted sum of the encoder hidden states:

$$\boldsymbol{c_i} = \sum_{j=1}^{N} \alpha_{ij} \boldsymbol{h_j} \tag{3.4}$$

Finally, the context vector $\boldsymbol{c_i}$, current decoder hidden state $\boldsymbol{s_i}$ and previous output y_{i-1} are used to compute the conditional probability as in Equation 3.1:

$$p(y_i | y_1 \ldots y_{i-1}, \boldsymbol{x}) = g(y_i, \boldsymbol{s_i}, \boldsymbol{c_i}) \tag{3.5}$$

where g is a non-linear, possibly multi-layered, function that outputs the probability of y_i.

Luong et al. [LPM15] abstracted the attention mechanism and described three different formulations for a scoring function between the current decoder hidden state $\boldsymbol{h_t}$ and each encoder hidden state $\bar{\boldsymbol{h}}_s$:

$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h_t^T} \bar{\boldsymbol{h}}_s & \textit{dot} \\ \boldsymbol{h_t^T} \boldsymbol{W_a} \bar{\boldsymbol{h}}_s & \textit{general} \\ \boldsymbol{v_a} \tanh(\boldsymbol{W_a}[\boldsymbol{h_t}; \bar{\boldsymbol{h}}_s]) & \textit{concat} \end{cases} \tag{3.6}$$

The *dot* attention computes a score between the decoder hidden state and an encoder hidden by computing the dot product between both vectors. In the *general* attention mechanism, the encoder hidden state is first multiplied by a learned weight matrix $\boldsymbol{W_a}$ before computing the dot product with the encoder hidden state. Finally, the *concat* attention mechanism works by concatenating the encoder and decoder hidden state into a single vector and multiplying it by a weight matrix $\boldsymbol{W_a}$. The resulting vector is passed through the tanh function, and finally the dot product is computed with a learned weight vector $\boldsymbol{v_a}$. The scoring function is then used for the computation of α_{ij} as opposed to e_{ij} in Equation 3.2:

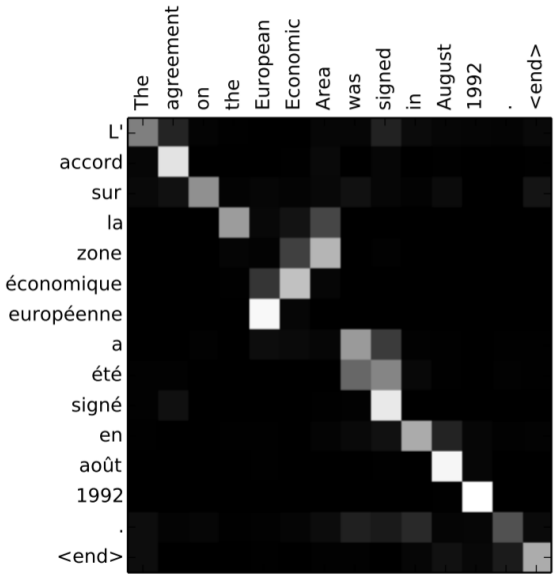

Figure 3.3: Alignment between an English source sentence (top) and the translated French target sentence (left) [BCB14]. Each pixel in row i and column j represents the attention weight α_{ij} on a greyscale (black: 0, white: 1) between target word x_i and source word y_j.

$$\alpha_{ij} = \frac{\exp(\text{score}(\boldsymbol{h}_i \bar{\boldsymbol{h}}_j))}{\sum_{k=1}^{N} \exp(\text{score}(\boldsymbol{h}_i \bar{\boldsymbol{h}}_k))} \tag{3.7}$$

The attention weights have the useful property that they can be easily visualized and used as a form of explanation for human users for *why* an NN model predicted a certain output. Applied to NMT, attention weights can be understood as a soft alignment between source and target sequences as seen in Figure 3.3. For each translated word, the weight distribution over the source sequence signifies which source words were most important for predicting this word.

3.3 Beam Search Decoding

Once an NMT model has been trained, it can be used for inference, e.g. translating a sentence from a source language to a target language. We are seeking the best prediction $\hat{\boldsymbol{y}}$ given the input sequence $\boldsymbol{x}$ (based on [HL17]):

$$\hat{\boldsymbol{y}} = \underset{\boldsymbol{y} \in \{\boldsymbol{y}^{|\boldsymbol{T}|}\}}{\text{argmax}}\, p(\boldsymbol{y}|\boldsymbol{x}) \tag{3.8}$$

where $\{\boldsymbol{y}^{|\boldsymbol{T}|}\}$ is the set of all possible output sequences for some maximum length T. Finding an optimal output by iterating through all possible output sequences is computationally unfeasible for large vocabularies, as there are $|\boldsymbol{V}|^{\boldsymbol{T}}$ possible such sequences. The simplest way of predicting a

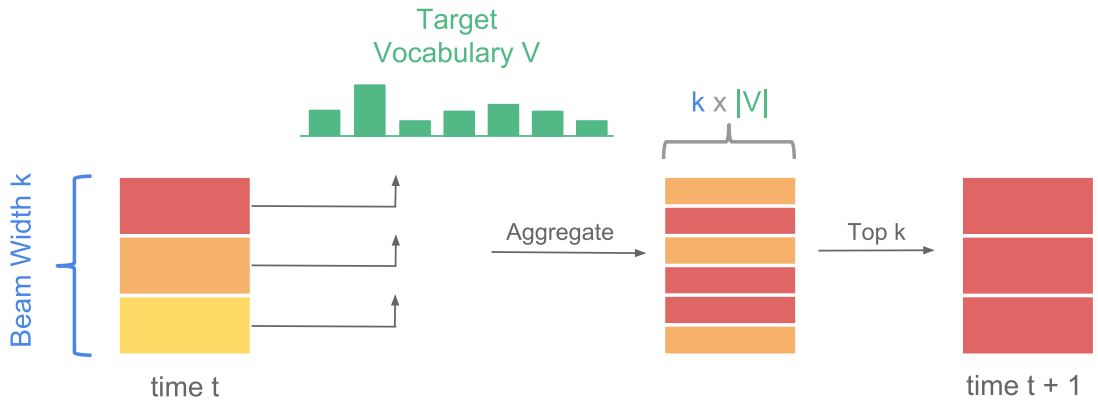

Figure 3.4: Transition from timestep t to $t + 1$ for beam search decoding with beam width k. At timestep t, k current hypotheses are in consideration. Each hypothesis is extended by computing the probability distribution over the target vocabulary of size $|\boldsymbol{V}|$ for the next token, resulting in $k \times |\boldsymbol{V}|$ possible extensions. The top k hypotheses are then chosen for the beam at timestep $t + 1$. Each box represents a hypothesis, with its probability mapped to the color of the box. Each stack of boxes represents a beam.

target sequence is greedy decoding, where at every time step, the token with the highest output probability y_t from the output vocabulary $\boldsymbol{V}$ is chosen as the next prediction and fed to the decoder in the next time step:

$$\hat{y}_t = \underset{y_i \in \boldsymbol{V}}{\operatorname{argmax}}\, p(y_i | \{y_1 \dots y_{t-1}\}, \boldsymbol{x}) \tag{3.9}$$

While greedy decoding is an efficient, straightforward way of generating an output sequence, it may become stuck in a local maximum. Consider an example sentence where a certain word has the highest probability of being the first output word, yet all possible output sentences starting with this word are unlikely, i.e. have a low average output probability. Then, first choosing a word with lower probability may lead to overall more likely output sentences.

Beam search decoding [Gra12] is a compromise between exhaustive search and greedy decoding. It enables to trade-off computation time in exchange for exploring a larger result space. In beam search decoding, a fixed number of hypotheses k is considered at each timestep, also called beam size or beam width. Each hypothesis is a (partial) output sequence, possibly ending with the EOS token. As seen in Figure 3.4, the model outputs a probability distribution of the next token over the target vocabulary for each hypothesis, resulting in $k \times |\boldsymbol{V}|$ possible hypotheses. These hypotheses are sorted by the probability of the latest token, and up to k hypotheses remain in the beam. Hypotheses ending with the EOS token are filtered out and put into the result set. Once k hypotheses are in the result set, the beam search concludes and the hypotheses are ranked according to a scoring function. Wu et al. [WSC+16] proposed the following scoring function s:

$$s(Y, X) = \frac{\log P(Y|X)}{lp(Y)} + cp(X, Y) \tag{3.10}$$

$$lp(X, Y) = \frac{(5 + |Y|)^{\alpha}}{(5 + 1)^{\alpha}} \tag{3.11}$$

$$cp(X, Y) = \beta * \sum_{j=1}^{|X|} \log \left(\min \left(\sum_{i=1}^{|Y|} \alpha_{ij}, 1 \right) \right) \tag{3.12}$$

where Y is a candidate translation, X is the source sequence and α_{ij} is the attention weight between the j-th source token and i-th target token. The length penalty lp normalizes the score with regard to length, as otherwise, shorter translations would be preferred due to their higher log probability, while the coverage penalty cp penalizes translations where source words do not receive enough attention. The parameter α with $0 \leq \alpha \leq 1$ controls the strength of length normalization, while β with $0 \leq \beta \leq 1$ controls the strength of coverage normalization. Setting $\alpha = 0$ and $\beta = 0$ will produce the same scoring function as during normal beam search decoding. Setting $k = 1$ reduces beam search to greedy decoding, as the single hypothesis will be extended be the most likely next token at each timestep.

A useful property of beam search decoding in regard to this work is its interpretability: hypotheses that are explored or discarded can be visualized and shown to end users, allowing a human translator to possibly pick a different hypothesis than predicted by the model. This might be useful because the scoring function in Equation 3.10 does not account for semantic meaning in the translation, which might be an important criterion for a user deciding between different translation hypotheses.

3.4 Handling Rare Words

One problem NMT models face is handling rare and unknown words. When a trained NMT model must translate a source sequence containing a word that was not encountered during training, that word will not be part of the source vocabulary V_s. The simplest solution to this problem is introducing a special unknown token (UNK), which acts as a stand-in for any word unknown to the model [SVL14]. During training, rare words, e.g. those who occur only once in the data set, are replaced with the UNK token, so that the model can learn to translate this token. The disadvantage of this approach is that translations generated by the model may contain the UNK token, requiring manual post-editing to replace them with the correct translation. This problem can be alleviated using the *copy mechanism* [APS16], whereby a source word is copied into the translation to replace a UNK token. The source word can be chosen by selecting the word from the source sequence that maximizes the attention weight of the target word. This approach can easily deal with unknown words such as names or numbers, where copying the word into the translation will lead to a correct translation but will likely fail to translate other types of words such as verbs.

Handling rare words in a better way is the main motivation for using *subword units* [SHB15] for NMT. Instead of choosing whole words for building the source and target vocabularies, words are split into subword units, which consist of one more characters. There are two main advantages of splitting up words in this way. First, the size of the vocabularies is reduced, since many rare words can be represented by more common subword units. For example, the words “translation” and “transport” may be segmented into “trans|lation” and “trans|port”. This helps to reduce model size and complexity, as well as training time. Second, the model can now handle any rare or unknown

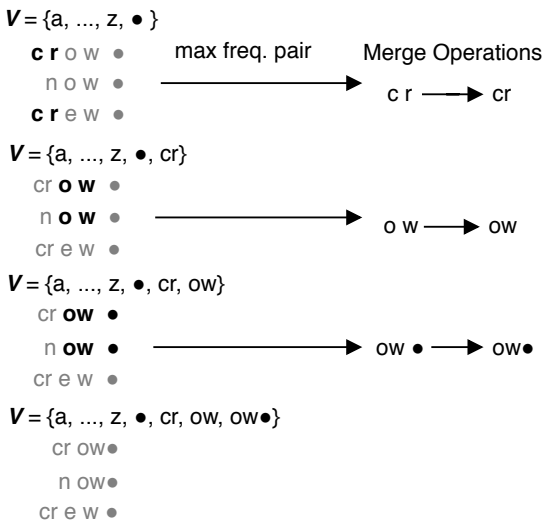

Figure 3.5: Three BPE merge operations on the words *crow*, *now* and *crew*. *V* is the vocabulary before each merge. The "●" symbol indicates the end of a word. After the final merge operations the words are segmented as *cr|ow*, *n|ow* and *cr|e|w*.

word by splitting it into its subword units, which are known beforehand and thus do not require a UNK token for translation. The question remains how to produce a segmentation of a word into subword units.

Byte Pair Encoding (BPE) [SHB15] is a method for compressing text by recursively joining frequent pairs of characters into a new symbol. This approach lends itself to be used for segmentation instead of compression. There are two distinct phases: learning BPE operations and applying BPE operations. For learning BPE operations, we start with a vocabulary containing only single characters that exist in the training data, and each word in the data split into single characters. Additionally, a special symbol "●" denoting the end of a word is added to the vocabulary in order to reverse the subword segmentation after the model has produced a translation. Then, the frequencies of all character pairs within word boundaries are computed. Finally, the most frequent character pair is merged into a single new symbol that is added to the vocabulary, with a symbol being a character n-gram. All occurrences of this pair in the data are also merged into the new symbol. The process is then repeated for a predetermined number of times. Figure 3.5 shows how merge operations are performed for a small example of three words. Note how in the first step, the character pair "n" and "c" are selected as the most frequent pair to be merged. After the first merge operation, the new symbol "cr" is added to the vocabulary, and the next merge operation can proceed. Also note that in the third operation, the most frequent pair is "ow" and "●", which shows that the "●" symbol is treated like any other character when counting frequent pairs. The number of merge operations is the only hyperparameter of this approach, and also controls the size of the resulting vocabulary, as each merge operation produces an additional symbol. For applying BPE operations, we start with an unsegmented word, split it into its constituent characters, and apply the merge operations that were previously learned. Once no more merge operations can be applied, the final segmentation is produced.

3.5 Domain Adaption

Documents often have a specific domain, e.g. legal, medical or scientific. Each domain has specific terminology, and the same word may even refer to different concepts in different domains. As such, the ability of NMT models to handle different types of domains is an important research topic. Domain adaption refers to techniques allowing NMT models trained on general training data, also called *out-of-domain*, to adapt to domain-specific documents, called *in-domain*. This is useful because while there may be an abundant amount of general training data, domain-specific data may be rare. Since NMT models need a large amount of training data to achieve good translation quality, the out-of-domain data can be used to train a baseline model, and the model can then be adapted using the in-domain data.

Techniques for domain adaption in NMT can generally be grouped into data centric approaches and model centric approaches [CW18]:

Data Centric approaches focus on the training data rather than changing the underlying model. Examples include the use of monolingual corpora or synthetic parallel data sets.

Model Centric approaches modify aspects of the model itself to adapt to a domain, such as specialized training objectives, NMT architectures or decoding algorithms.

Fine tuning is an example of a model centric approach focusing on the training objective. First, a general model is trained on a large set of out-of-domain data. The model is then fine-tuned on in-domain data, which typically contains a smaller number of sentences. This reduces the problem of training an NMT model in a low-resource scenario where little parallel data sets exist for a given domain. During fine-tuning, a subset of model weights is often frozen, e.g. such that only the weights of the output layer are adjusted for the in-domain data.

3.6 Training

The general mechanism for training a NMT model is the same as training any ordinary NN model as described in Section 2.2.2. Let $D = \{(\boldsymbol{x}^t, \boldsymbol{y}^t)\}_{t=1}^{T}$ be a parallel corpus of size T, where $\boldsymbol{x}^t$ and $\boldsymbol{y}^t$ are source and target sequences, respectively. We are seeking a model $p_{\boldsymbol{\theta}}$, parametrized by the weights $\boldsymbol{\theta}$, that minimizes a loss function $L_{\boldsymbol{\theta}}$, typically chosen as the negative log-likelihood [PCC17]:

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, L_{\boldsymbol{\theta}} \tag{3.13}$$

$$= \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{t=1}^{T} -\log(p_{\boldsymbol{\theta}}(\boldsymbol{y}^t|\boldsymbol{x}^t)) \tag{3.14}$$

$$= \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{t=1}^{T} \sum_{i=1}^{N_i} -\log(p_{\boldsymbol{\theta}}(y_i^t|y_0^t, \ldots, y_{i-1}^t, \boldsymbol{x}^t)) \tag{3.15}$$

where N_i is the length of the i-th target sequence. This formulation follows directly from the decomposition in Equation 3.1, as taking the log of a product is equal to the sum of log values. Then Stochastic Gradient Descent (SGD) is typically used to iteratively find the optimal weights $\hat{\boldsymbol{\theta}}$.

3.7 Evaluation Metrics

Evaluation metrics are important for all ML tasks, as they allow to compare models regarding their performance for a certain task on a quantitative level. An evaluation metric can be seen as any function that takes as input (1) the outputs of the model to be evaluated and (2) ground truth representing the real outputs and computes a score that represents the performance of the model regarding a certain task. Often, multiple evaluation metrics for a given task exist, as their use may depend on context or because a single metric might not represent all aspects of model performance.

One issue in machine translation is the subjective nature of translations: there is rarely a single correct translation for a given source sentence, and multiple translations might be considered as correct by different people. Since translations are ultimately made for human understanding, subjective ratings of professional translators would be the best possible evaluation metric, but since using humans for evaluation is slow and costly, metrics that can be computed automatically are necessary. Due to these problems, it is difficult to formalize and quantify the quality of translations generated by an NMT model. Nonetheless, several evaluation metrics have been proposed, focusing on capturing different aspects of translation quality.

The Bleu metric [PRWZ02] is based on n-gram precision and takes as input for each source sentence (1) a candidate translation of the model that is to be evaluated and (2) a list of reference translations, e.g. translations of multiple human translators. Specifically, a modified n-gram precision p_n for $n \in \{1, \ldots, 4\}$ is computed, where the n-gram count is clipped to the maximum number of times it appears in any reference translation. This clipping is necessary as otherwise, candidate translations containing multiple repetitions of a word may have high precision values, thus distorting the overall score. The precision is then calculated as the ratio between the clipped counts and all counts of each n-gram in the candidate translation. Finally, a Brevity Penalty (BP) is applied to the overall score of a document, which penalizes candidate translations that are shorter than any reference translation. Putting it all together, the Bleu score is computed as follows:

$$\text{Bleu} = \text{BP} \cdot \exp\left(\sum_{n=1}^{4} \frac{1}{4} \log p_n\right) \tag{3.16}$$

Bleu has been shown to correlate well with the judgment of professional human translators. It is important to note that Bleu is a corpus-level metric: while scores on individual sentences might differ greatly from human translators, averaging over all sentences of a corpus produces a useful score of translation quality. The Bleu score has become one of the most used metrics for evaluating NMT models and is therefore useful to compare our system to state-of-the-art NMT models.

Translation Edit Rate (Ter) [SDS+06] is yet another evaluation metric. Ter considers the minimum number of edits necessary to convert a candidate translation into one of the reference translations. Thus, it can be seen as a measure of how much effort a human translator would have to expend to post-edit a machine-generated translation into a reference translation. The Ter score is computed as the ratio of the number of edits and the average length of the reference translations to normalize with regards to sentence length. The edits considered for Ter are insertion, deletion, and substitution of single words as well as shifts of word sequences, and all edits have a uniform cost. This differentiates Ter from Bleu, since it does not consider n-gram overlaps. Ter is also case-sensitive and considers

punctuation for edits. The main advantage of TER is its intuitiveness, as it simply measures the effort needed to transform a candidate translation into a reference translation. A disadvantage of TER is its high computational complexity compared to BLEU.

CharacTER [WPRN16] is a character-level extension of TER. Like TER, CharacTER first calculates shift edits on word level but uses character-level edit distance on individual words. CharacTER is found to correlate better with human judgments for morphologically rich languages on a system level since it takes character-level differences of words into account. It also outperforms other metrics, including BLEU and TER, on standard data sets, especially for German → English and English → German translation. It can also be applied on sentence level as opposed to BLEU. We use CharacTER for automated evaluation of translations generated by the NMT model to analyze correlations for translation quality.

4 System Description

This chapter introduces NMTVis (**N**eural **M**achine **T**ranslation **Vis**ualization), a VA system for NMT. First, an overview of the most important goals and requirements for the system is given, followed by detailed descriptions of the different views and visualizations, implementation details, and the NMT model that backs the system.

4.1 Goals & Requirements

This chapter will outline the goals and requirements that have to be satisfied by the system. In order to build an effective system, we must first define the goals a system wants to achieve, the context under which a system operates and define tasks that help to achieve the goals considering the context. We will use the conceptual framework from Hohman et al. [HKPC18] for analysing VA approaches for Deep Learning to contextualize our approach. Answering the six questions, this work can be summarized as follows:

> To explore translation hypotheses and correct translation errors (**why**), professional translators (**who**) visualize attention weights and beam search trees of an NMT model (**what**) using dynamic graph visualizations (**how**) after the training phase (**when**) to aid in the translation of large, domain-specific documents (**where**).

The following goals were identified, based on the previous description:

G1 Help translators to translate large, domain-specific documents: The main goal of the system is to help users, specifically translators, to translate large, domain-specific documents efficiently. The main assumption of our system is that the translator is given the task of translating a large, domain-specific document while facing time constraints, meaning that not every sentence can be manually translated or post-edited in detail.

G2 Recommend users critical sentences: The most critical sentences, i.e. sentences whose correction will likely lead to improved translation quality in the remaining sentences, should be recommended to the user to decrease overall post-editing workload.

G3 Interactively correct translations: The user should be able to see how the model arrived at a translation, and interactively explore different hypotheses in collaboration with the model. This should relieve the burden of purely manual translation while also benefiting from a user's domain knowledge. Corrections of any kind to the translation should be supported and applying them should be simple and efficient.

G4 Feed corrections back into the model and improve translations: The corrections made by the user should be fed back into the model and possibly improve remaining translations, freeing the user of the burden to correct large amounts of text manually.

The goals explain what we want to achieve with the system but not how we can achieve these goals. Therefore, tasks were derived from the goals, with a task being a description of functionality of the system from the point of view of a user. The following tasks were identified, with each task contributing to achieving one or more of the goals:

T1 Show an overview of a document and its sentences as well as translations made by the NMT model to give an overall context during post-editing.

T2 Find, filter and select critical sentences for post-editing based on relevant metrics such as translation confidence, sentence length, and number of keyphrases.

T3 Visualize translation hypotheses as generated by the model during beam search decoding to aid the user during post-editing.

T4 Explore and edit hypotheses using interactive visualizations to correct incorrect translations generated by the system.

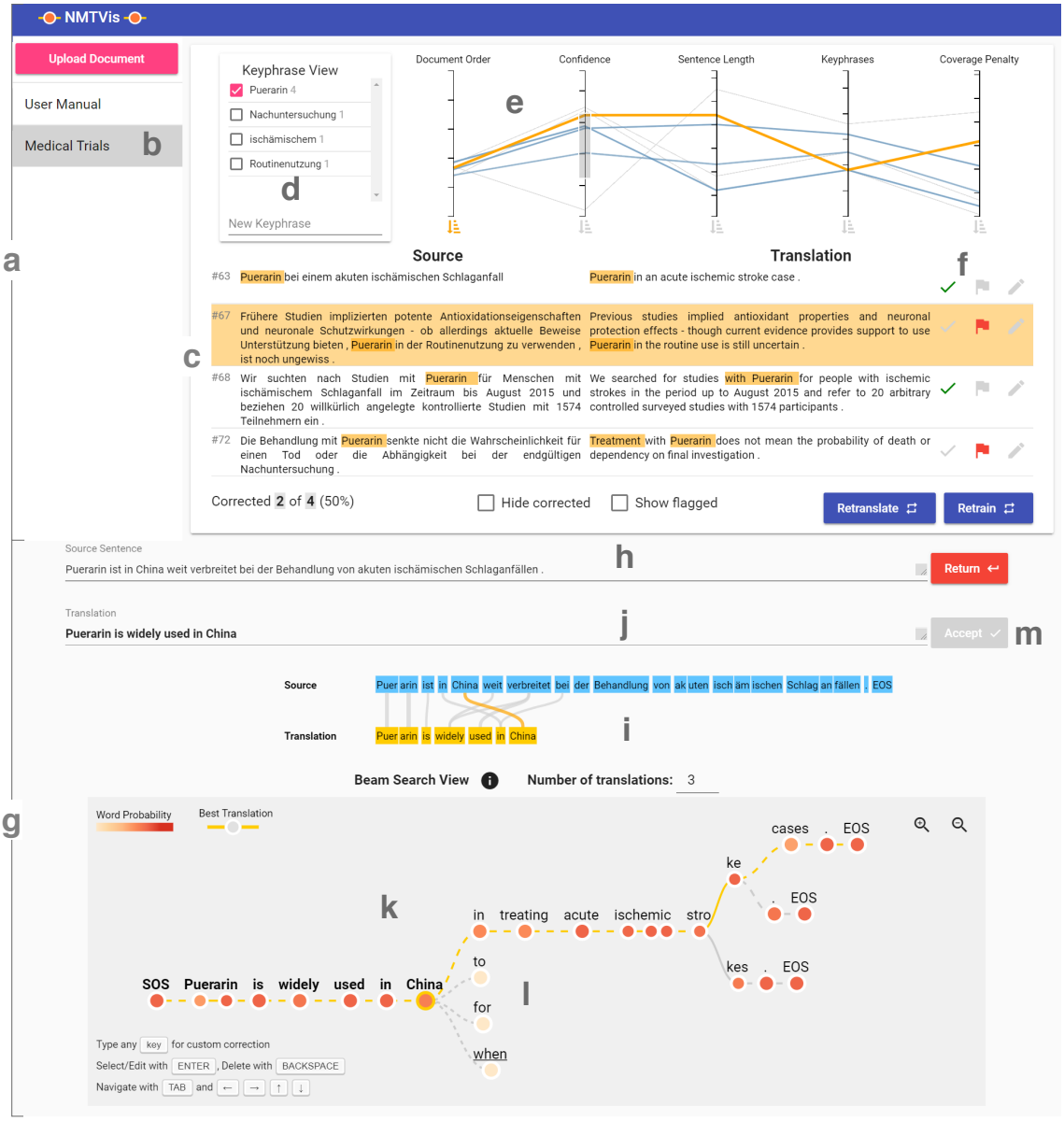

Figure 4.1: The NMTVis user interface: The user opens the Document View (a) and selects a document for translation (b). Source sentences and machine translations are displayed side by side (c). Domain-specific words can be filtered in the Keyphrase View (d) or based on metrics in the Metrics View (e). Translations can be directly accepted or flagged for later correction (f). Detailed post-editing and in-depth analysis of a single sentence are conducted in the Sentence View (g). The source sentence is displayed on top for context (h). The Attention View (i) visualizes attention weights for the current translation (j). The Beam Search View (k) shows a tree of translation hypotheses that can be interactively explored and corrected (l). Finally, the user can accept the post-edited translation to adapt the model to the current document (m).

4.2 Workflow

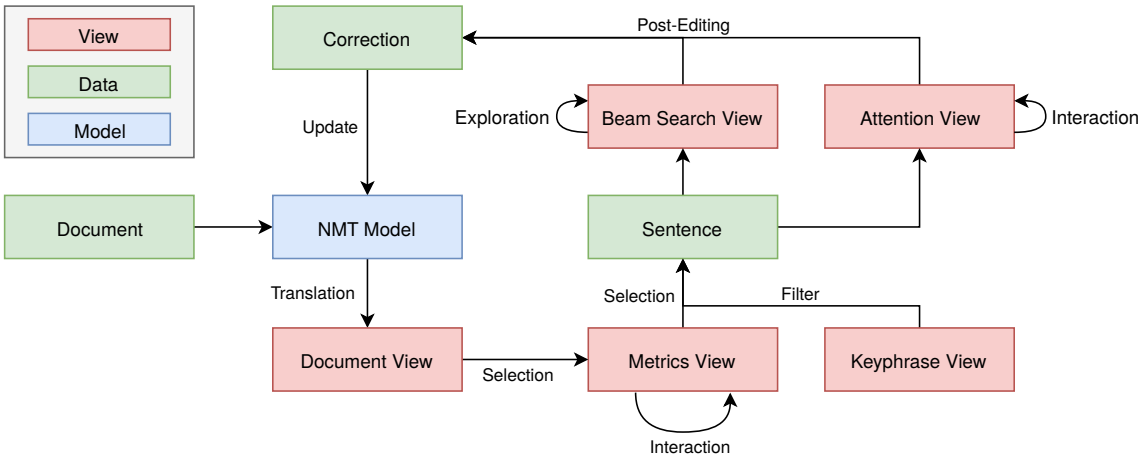

Figure 4.2: A visual representation of the translation workflow of NMTVis. Boxes represent views, models and data. Arrows represent the flow of data and transitions between different views. Arrow labels describe actions by the system and user.

Before describing the different views and components of NMTVis, the standard translation and post-editing workflow will be briefly described, relating how the different views and components work together to achieve the goals outlined before. Figure 4.2 shows a visual representation of the standard workflow in NMTVis. Documents are uploaded by the user, and translated by the NMT model. The source sentences and translations are shown in the Document View after the user selects a document. Once a document is selected, the user can browse the document and use the Metrics View to find sentences to correct. Using the Keyphrase View, the sentences can be filtered to analyze the system's translation regarding specific terminology. The user can flag machine-generated translations for later correction. Once a sentence is selected, the user opens the Beam Search View and the Attention View to interactively explore alternative translations and post-edit the machine translation. When the user is finished with post-editing, their correction is fed back to update the NMT model. This feedback loop is the core of the translation process, allowing the user to collaboratively work with the NMT model. Note how this resembles the standard VA process as described in Section 2.6.2, where a feedback loop is also formed by a user gaining knowledge about data and models through interaction with visualizations. Figure 4.1 gives a preliminary overview over system components and views and how they relate to the workflow.

4.3 Document View

The *Document View* (Figure 4.3), related to task **T1**, is responsible for managing user-provided documents and showing an overview of source sentences and translations for a given document. Machine-generated translations can be accepted or flagged for later correction from the sentence list, which enables a fluent post-editing workflow where multiple sentences can be reviewed in sequence. Sentences can be filtered to either hide already corrected translations or show only translations that were flagged. The Document View enables domain adaption by allowing users to (1) retrain

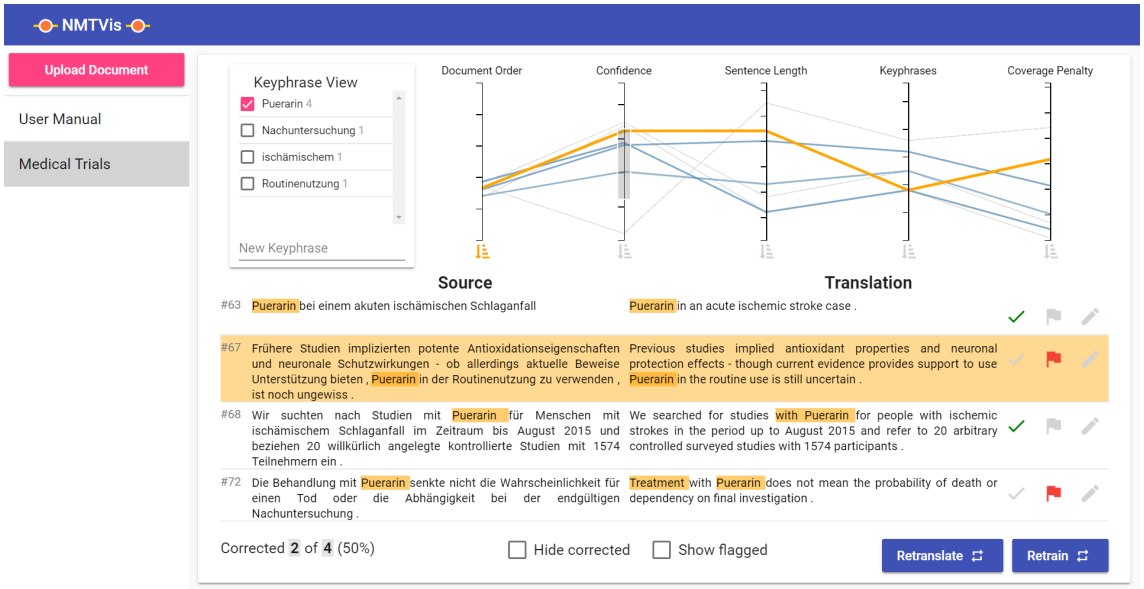

Figure 4.3: The Document View in detail: Users can upload new documents are view existing ones (left). The Keyphrase View and Metrics View are embedded for the currently selected document (top). Three actions can be performed by clicking the respective icon next to a translation: (1) marking a translation as correct (2) flagging a translation for later correction and (3) editing a translation in a separate view. The current translation progress is displayed on the bottom left. Filters (bottom middle) allow users to hide corrected translations and to show flagged translations. Finally, the underlying NMT model can be retrained and uncorrected translations can be re-translated (bottom right).

the NMT model based on corrections and (2) re-translate uncorrected sentences. In the second case, the changes between the previous translations and the new translations are also shown in the sentence list. This allows users to quickly see how retraining changed the translations.

New documents can be uploaded to the Document View, that are then translated by the NMT model and shown in the document list on the left. Each uploaded document is run through a preprocessing pipeline to allow translation by the NMT model. First, the document is separated into separate sentences, as NMT models translate on sentence-level. Then, a tokenizer splits each sentence into tokens for further processing. BPE is applied to the tokenized sentences using the same merge operations as learned from the training data. Afterward, each sentence is translated by the NMT model using beam search decoding. Finally, the list of translation hypotheses as generated by beam search decoding is then saved along with attention weights for each hypothesis and for each sentence.

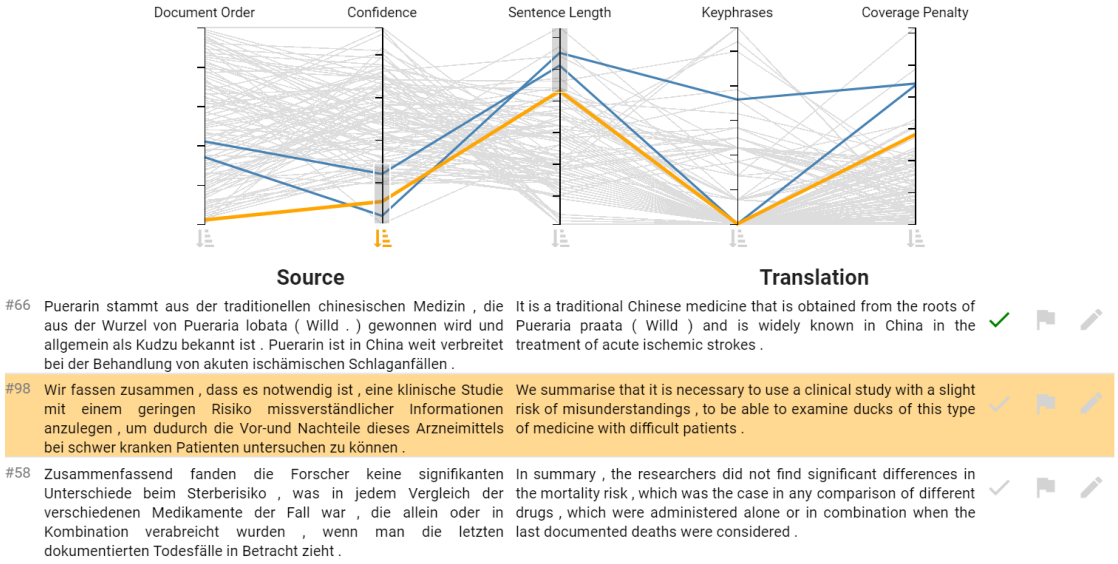

Figure 4.4: In the Metrics View, each sentence is visualized as a line in a parallel coordinates plot. Blue lines represent currently visible sentences, grey lines represent sentences that are filtered out. Each axis in the plot represents a metric based on the source sentence and translation. Sentences in the Metrics View and the list below are linked so that highlighting a sentence (orange highlight) updates both views. Metrics can be reordered by dragging and dropping and used for sorting. In this example, the document is sorted in ascending order based on confidence, and two filters for low confidence and high sentence length are active.

4.4 Metrics View

4.4.1 Description

The *Metrics View* (Figure 4.4), related to task **T2**, shows sentence scores for evaluation metrics to enable the user to quickly find sentences that are likely to contain translation errors. As there are multiple relevant metrics per sentence, the Metrics View needs to support visualization of multivariate data. Parallel coordinate plots [Ins85] are a common visualization technique for handling such data. They are useful for analyzing correlations between neighboring dimensions, as well as finding outliers and general trends. Each sentence is mapped to a line in the plot, and the y coordinates at each axis are determined by the value of the metric of the respective sentence. To enable comparison between any two metrics, the user can drag and drop each to change the metrics' order. Brushing and linking is also possible, with each brush acting as a filter of the currently displayed sentences below. This can be achieved by clicking and dragging on the axis of a metric, creating a brush over the respective axis. This brush can also be slid up and down the axis by clicking and dragging it to shift the current filter for a given metric. Hovering over a sentence in the text area will also highlight the relevant line in the parallel coordinates plot to allow the user to see the metric scores for that sentence at a glance.

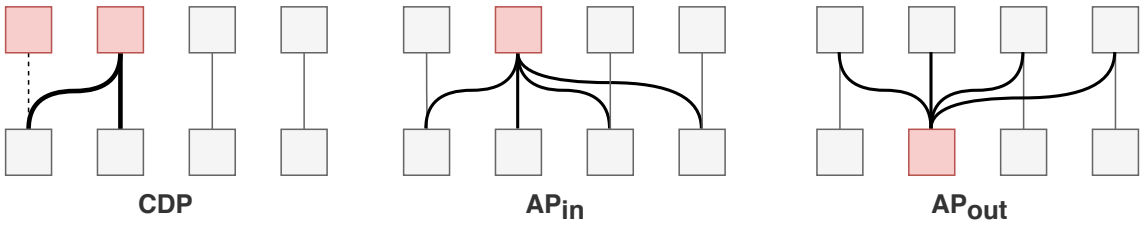

Figure 4.5: Illustration showing three cases for high CDP, AP_{in} and AP_{out} scores. Boxes represent input and output tokens. Lines represent attention weights, with thick lines being large attention weights and dashed lines being missing attention weights.

4.4.2 Metrics

The following section will introduce the metrics used in the Metrics View. The attention matrix is denoted as $\boldsymbol{\alpha}$ where each entry α_{ji} is the attention weight between source word x_i and target word y_j with source sequence X and target sequence Y. Coverage Penalty (CP), as introduced by Wu et al. [WSC+16], penalizes translations where input tokens are not given enough attention:

$$\mathrm{CP}(\boldsymbol{\alpha}) = -\sum_{j=1}^{|X|} \log\left(\min\left(\sum_{i=1}^{|Y|} \alpha_{ji}, 1\right)\right) \tag{4.1}$$

We will now introduce the following attention-based metrics as suggested by Rikters et al. [RFB17]:

Coverage Deviation Penalty (CDP) penalizes excessive or deficient attention per input token.

Absentmindedness Penalty (AP) penalizes scattered attention either per input (AP_{in}) or output token (AP_{out}).

Confidence aggregates the previous metrics into a single score.

CDP penalizes excessive attention per input token. Input tokens whose summed attention is higher or lower than 1 have high CDP scores, while input tokens with a summed attention of 1 have a score of 0:

$$\mathrm{CDP}(\boldsymbol{\alpha}) = \frac{1}{|X|}\sum_{j=1}^{|X|} \log\left(1 + \left(1 - \sum_{i=1}^{|Y|} \alpha_{ji}\right)^2\right) \tag{4.2}$$

The final CDP score is then normalized by the length of the source sequence. AP penalizes scattered attention either per input (AP_{in}) or output (AP_{out}) token. AP_{in} is high then an input token is paid attention to by many output tokens, while AP_{out} is high when an output token pays attention to many input tokens.

$$\mathrm{AP}_{\mathrm{in}}(\boldsymbol{\alpha}) = -\frac{1}{|Y|}\sum_{j=1}^{|X|}\sum_{i=1}^{|Y|} \alpha_{ij}\log(\alpha_{ij}) \tag{4.3}$$

$$\mathrm{AP}_{\mathrm{out}}(\boldsymbol{\alpha}) = -\frac{1}{|Y|}\sum_{i=1}^{|Y|}\sum_{j=1}^{|X|}\alpha_{ji}\log(\alpha_{ji}) \tag{4.4}$$

This metric is based on the cross entropy measure of a probability distribution, which measures the randomness or scatteredness of a distribution. Attention matrices that distribute attention equally among tokens will have high AP values, while distributions that distribute it only to a few tokens will have low AP values. Note that to compute $\mathrm{AP}_{\mathrm{in}}$, attention values must first be normalized by the summed attention per input token. Finally, these metrics are combined into a single *confidence* score, which sums up CDP, $\mathrm{AP}_{\mathrm{in}}$ and $\mathrm{AP}_{\mathrm{out}}$:

$$\mathrm{Confidence}(\boldsymbol{\alpha}) = e^{-0.05(\mathrm{CDP}(\boldsymbol{\alpha})+\mathrm{AP}_{\mathrm{in}}(\boldsymbol{\alpha})+\mathrm{AP}_{\mathrm{out}}(\boldsymbol{\alpha}))} \tag{4.5}$$

Adding these metrics into a single confidence score increases the robustness of the resulting metric, as each metric focuses on a different aspect of the attention distribution. Figure 4.5 illustrates the characteristics of the metrics through examples. For CDP, the first input token has a missing attention weight from the first output token, while the second input token has high attention weights from the first and second output token. Therefore, the attention weights summed over all output tokens deviate from 1 for the first two input tokens, resulting in a high CDP score. In the second example, all output tokens pay attention to the second input token, resulting in a scattered attention distribution. This results in a high $\mathrm{AP}_{\mathrm{in}}$ score. In the third example, the second output token pays high attention to all input tokens, thus resulting in high $\mathrm{AP}_{\mathrm{out}}$, as the attention is spread over multiple input tokens.

Other Metrics

From the introduced metrics, CP and confidence are used in the Metrics View. We did not add the CDP, $\mathrm{AP}_{\mathrm{in}}$ and $\mathrm{AP}_{\mathrm{out}}$ metrics to the final Metrics View, as they are aggregated into the confidence score, and thus do not offer useful additional information. Additionally, the following non-attention based metrics are used in the Metrics View:

Sentence Length: The number of words in a source sentence. This enables users to explore outliers such as extremely short or long sentences. Long sentences are also more likely to contain translation errors due to the tendency of NMT model to perform worse on long sentences, making them interesting for closer analysis.

Document Index: The index of a sentence in the original document, starting from 0 for the first sentence. Sorting by the document index allows users to see sentences in their original order, which can be important to for context during post-editing. Furthermore, this metric may reveal certain trends for a document, e.g. if multiple sentences in a row have low confidence scores or if certain keyphrases are predominantly used in specific parts of a document.

Keyphrases: The number of keyphrases in a source sentence weighted by the frequency of each keyphrase occurring in a source sentence. This metrics allows to filter and sort by sentences that contain many domain-specific words. This is important as the NMT model may produce erroneous translations for such words, as they have low frequency in the out-of-domain training data.

4.4.3 Usage Scenarios

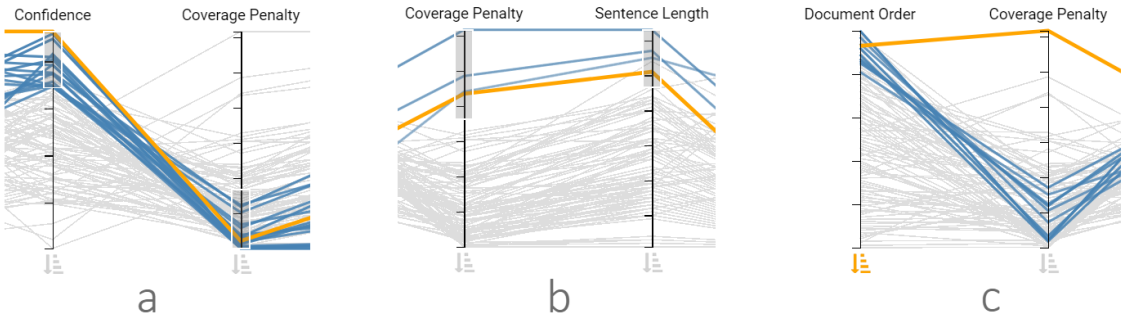

Figure 4.6: Three usage scenarios for the Metrics View: (a) A negative correlation between neighboring metrics is uncovered. (b) Outliers for neighboring metrics are found. (c) A cluster of sentences and an outlier are found for an active keyphrase (not depicted).

Figure 4.6 shows three common scenarios when using the Metrics View. The first scenario depicts a correlation between neighboring metrics. Here, high confidence scores seem to correlate with low coverage penalty, as seen by the diagonal lines crossing between both metric axes. Creating a filter for high confidence reveals a cluster of sentences with low coverage penalty and vice versa. Uncovering these correlations can be useful to detect erroneous translations. For example, if confidence negatively correlates with coverage penalty, filtering for sentences with both low confidence and high coverage penalty should be more robust than filtering only based on either metric.

The second scenario is outlier detection. As the parallel coordinates plot makes sentences where any metric has extremely low or high values visible, outliers can easily be detected. Here, four sentences with significantly higher coverage penalty when the remaining sentences are seen. Applying a filter for high coverage penalty also reveals that these sentences have high sentence length, suggesting that the system makes more translation errors for long sentences. Having found these outliers, they can then be analyzed in the list of sentences below.

The third scenario is a combination of the first two. By activating a keyphrase, a cluster of sentences with low document order is revealed. This suggests that this keyphrase appears at the beginning of the document. Also, a negative correlation with coverage penalty is seen, implying that this keyphrase was likely translated accurately for most sentences. At the same time, a single outlier with high coverage penalty is seen, which can then be selected for manual review by hovering over the respective line in the Metrics View.

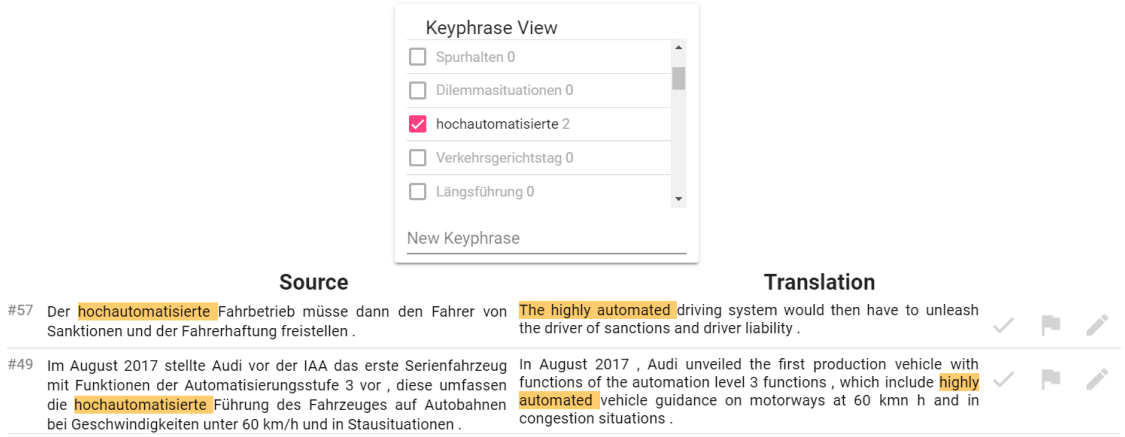

Figure 4.7: The Keyphrase View displays the list of keyphrases automatically extracted from the current document. The number of currently filtered sentences in the document containing each keyphrase are displayed on the right. A text input on the bottom allows users to quickly enter new keyphrases. Each keyphrase occurrence is highlighted in the source sentences below, as well as target words with attention weight above a threshold to indicate how the keyphrase was translated.

4.5 Keyphrase View

4.5.1 Description

The *Keyphrase View*, related to task **T2**, shows a list of keyphrases that were automatically extracted from the current document. The main use of the Keyphrase View is sentence selection based on keyphrases. This allows users to check if domain-specific words were translated correctly by the system and find sentences with erroneous translations efficiently. Keyphrases are extracted from a document based on their frequency in the general out-of-domain data the NMT model was trained on, as well as their frequency in the document itself. Candidate words are first extracted from the document by removing stop words, and for each candidate its frequency, i.e. the number of occurrences in the out-of-domain data is determined. Then, all candidates whose frequency is below a predefined threshold are selected as domain-specific keyphrases and sorted in descending order based on their frequency in the in-domain document.

There are multiple ways of interacting with the Keyphrase View. First, keyphrases can be toggled on or off by clicking on its related checkbox. Toggling on a keyphrase makes it active, filtering the current sentences based on whether a sentence contains the given keyphrase. If multiple keyphrases are active, then a sentence has to contain all active keyphrases to be displayed, which allows users to quickly find sentences related to certain topics. The number next to each keyphrase indicates the number of sentences this keyphrase occurs in for the currently filtered sentences. Keyphrases that do not occur in the currently filtered sentences are disabled.

Hovering over an inactive topic acts as a temporary filter for the Metrics View: sentences that do not contain that topic are greyed out so that the user can get an overview which sentences are affected by activating that topic. This feature also allows users to detect patterns in the Metrics View for a keyphrase e.g. if most sentences of a topic have high confidence values, then the model can most

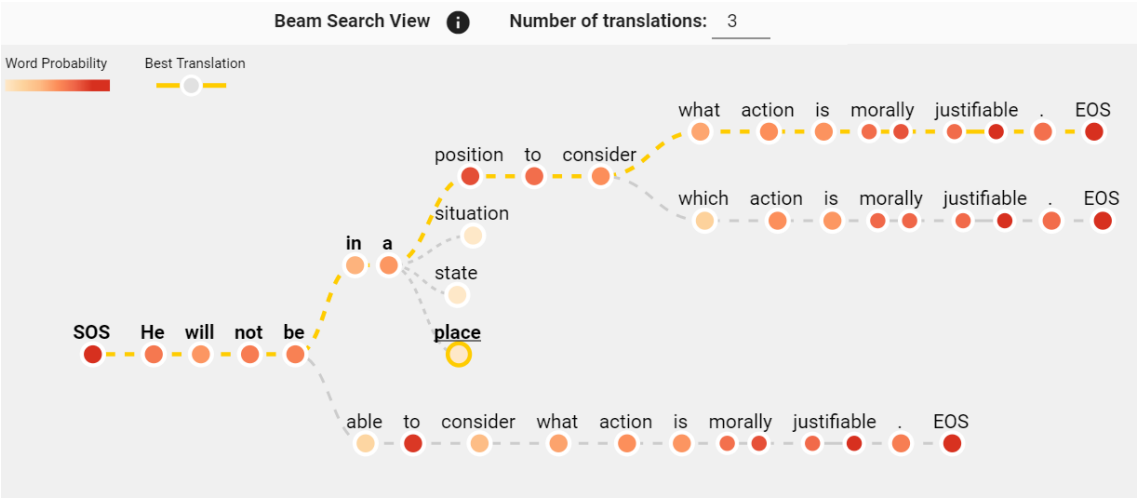

Figure 4.8: The Beam Search view visualizes translation hypotheses found during beam search decoding as a tree structure. Each node represents a single output word, with word probability given by the NMT model mapped to the node color. A path from the SOS root node to an EOS leaf node represents a complete translation hypothesis. Highlighted edges signify the current best hypothesis.

likely translate sentences containing this keyphrase well. New keyphrases can be added manually by the user at any time, which enables filtering based on words that were not automatically extracted from the document.

Additionally, words containing an active keyphrase are highlighted in the source sentence to get an overview at a glance. More importantly, the words in the target translation are highlighted based on the soft alignment of the attention weights as seen in Figure 4.7. By visualizing which target words were translated from the highlighted source words, the user can see how domain-specific words were translated by the model and whether there are systematic translation errors for certain keyphrases.

4.6 Beam Search View

This section will introduce and discuss the Beam Search View, including its data model, visual encoding and interaction concept.

4.6.1 Description

The *Beam Search View* (Figure 4.8), related to tasks **T3** and **T4**, visualizes translation hypotheses generated by beam search decoding and allows users to interactively explore and adjust each hypothesis. As was done in related works [SGB+18] [LSK17], we chose to visualize beam search hypotheses as a tree structure due to the inherently hierarchical nature of the decoding process. The main advantage of this approach is that multiple hypotheses can be compactly shown at once,

compared to displaying a flat list of hypotheses. Multiple interaction techniques allow exploring alternative translation hypotheses and incorporate custom user corrections to the Beam Search View.

4.6.2 Data Model

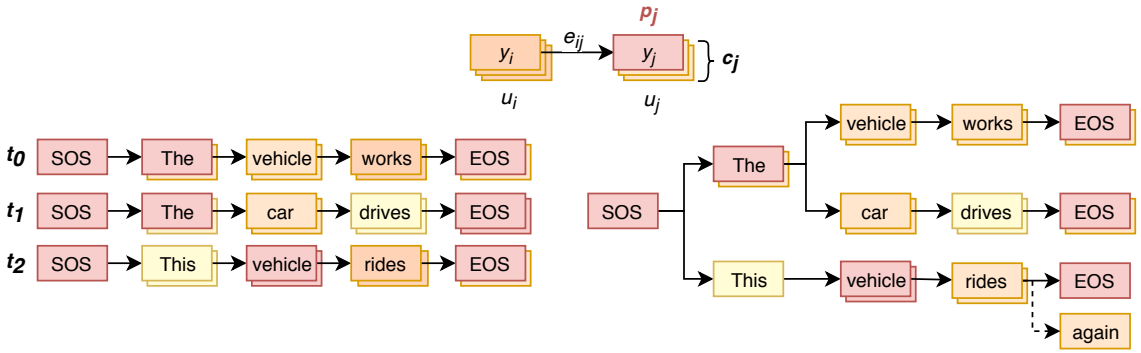

Figure 4.9: Illustration of the data model (top) of the Beam Search View. Nodes u_i and u_j are connected through edge e_{ij}. Node u_j has associated word y_j, output probability p_j and candidate set $\boldsymbol{c_j}$. Initial translation hypotheses generated by beam search decoding (left) and transformed tree structure used for Beam Search View (right). Each word (box text) is associated with a probability (box color) as generated by the NMT model, and a set of alternative candidate words (hidden boxes).

We will now introduce the data model that backs the Beam Search View. Let $\boldsymbol{H} = \{t_1, \ldots, t_N\}$ be the initial set of N translation hypotheses generated by an NMT model p during beam search decoding from a source sequence $\boldsymbol{x}$. Each hypothesis t_i is defined as a sequence of tuples $d_j = (y_j, p_j, \boldsymbol{c_j})$:

$$t_i = \{(y_1, p_1, \boldsymbol{c_1}), \ldots, (y_{n_i}, p_{n_i}, \boldsymbol{c_{n_i}})\} \tag{4.6}$$

where n_i is the length of the translation hypothesis, y_j is the j-th target word and p_j its output probability $p(y_j|y_1, \ldots, y_{j-1}, \boldsymbol{x})$ and $\boldsymbol{c_j}$ is a set of candidate words that were pruned during beam search decoding. By definition, y_1 is always the special SOS token with $p_1 = 1$ and $\boldsymbol{c_1} = \{\}$ indicating the start and y_{n_i} the EOS token indicating the end of a sequence. Then, $\boldsymbol{H}$ is transformed into a directed tree $\boldsymbol{T} = (V, E)$ with nodes V and edges E, by merging common prefixes of hypotheses. Each node is associated with a tuple $d_j = (y_j, p_j, \boldsymbol{c_j})$ as before. The root node $r \in V$ is associated with the SOS token, since every translation hypothesis starts with it by definition. Additionally, leaf nodes of the tree are associated with EOS tokens, because translation hypotheses end with the EOS token. Let $(r, \ldots, l)$ be a path from the root node r to a leaf node l. A complete translation hypothesis in $\boldsymbol{T}$ is defined by the path from root node r to l by concatenating the associated d_j of each node $v_j \in V$ in the path. The tree $\boldsymbol{T}$ serves as the backing model of the Beam Search View.

Figure 4.9 illustrates the data model and the transformation from a set of translation hypotheses $\boldsymbol{H}$ to a tree of hypotheses T on three sample hypotheses. For each element the tuple $d_j = (y_j, p_j, \boldsymbol{c_j})$ is shown as a box, where the probability p_j is represented by the color of each box, the word shown is y_j, and the set of candidates c_j are represented by the partially hidden boxes. On the right, the

Figure 4.10: The visual encoding used in the Beam Search View graph. **Word Boundary:** Two separate words are connected through a dashed, thin line. **Subword Boundary:** Subwords within a word are connected through a continuous, thick line. **Word Selection:** Selected (sub)words are bolded, the border around the word's node is highlighted in orange. **Hypothesis Selection:** All lines connecting (sub)words within the selected hypothesis are highlighted in orange.

transformed tree $\boldsymbol{T}$ is shown. Note how the first two hypotheses both start with the prefix "SOS The", and are therefore merged up to the last common prefix word in the tree. Thus, the tree is more compact compared to the list of hypotheses, as all common prefixes of the hypotheses are shared.

4.6.3 Visual Encoding

Figure 4.10 shows the visual encoding used for the Beam Search View. At a basic level, words are mapped to tree nodes, and neighboring words in a hypothesis are connected through links. As detailed in Section 3.4, subword units are an important technique for NMT systems to solve the rare word problem. Therefore, our visualizations must also support subwords and handle them intuitively for the user. In order to visualize the difference between normal words in a translation hypothesis, and subword units that form a single word, we map the type of word boundary to the visual property of the link between (sub)words. Ordinary word boundaries are represented by thin, dashed lines, while subword boundaries are visualized by thick, continuous lines. Additionally, the inter-node distance between subwords is decreased compared to ordinary words to further signalize that connected subwords form a single unit. While it might seem reasonable to merge neighboring subword nodes into a single word node, this would make it impossible to explore hypotheses that branch off of a compound word consisting of subword units by interacting with individual nodes. Concerning node interaction, two additional visual mappings are applied to nodes and links. First, the border of (sub)word nodes that are selected by clicking or hovering are highlighted, and the word text is bolded and enlarged. Second, upon selecting an EOS node, all links belonging to the respective translation hypothesis are highlighted. This enables the user to see the current best translation at all times for context.

Each node has two parts, the actual node visualized as a colored circle, and the word text centered above the circle. The circle's color is a mapping from the word's output probability. This helps the user to see which parts of translation hypotheses might need further manual review, as the NMT model predicted words with low probability. Because the probability of a word can be seen as an uncertainty of the NMT model regarding its prediction, results from uncertainty visualization were used to decide on the visual mapping. Many visual variables can be used to express uncertainty, such as fuzziness, location, color value, hue or size. MacEachren et al. [MRO+12] evaluated these variables in an empirical study regarding their intuitiveness, accuracy, and precision. For this work,

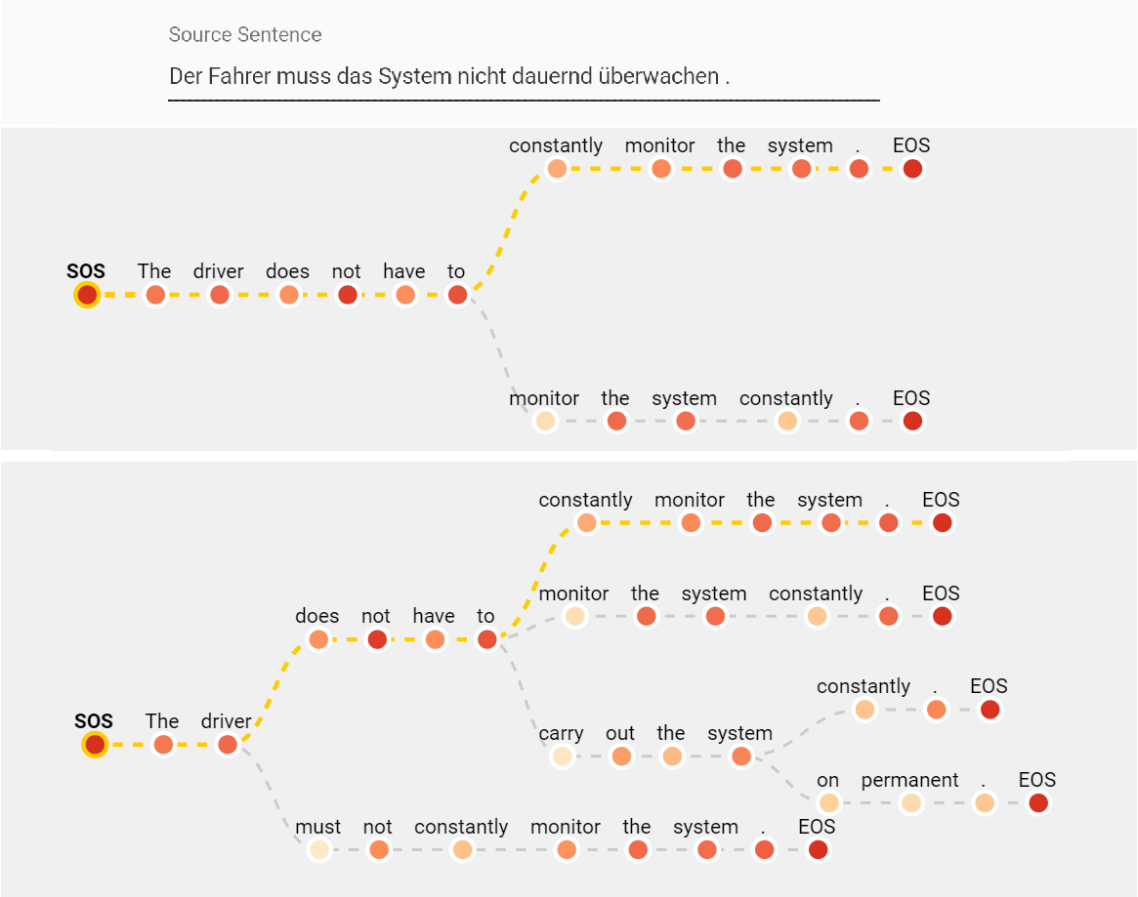

Figure 4.11: The Beam Search View with a varying number of translations hypotheses. A sample source sentence (top) is translated into two translation hypotheses (middle) and into five translation hypotheses (bottom). A higher number enables exploration of more diverse translations at the expense of increased visual clutter.

intuitiveness is most important, as the uncertainty should simply inform the user, and the exact probability is not important. Therefore, we selected color value as our visual variable to visualize word probability.

The user can manually increase the beam size to explore a larger space of hypotheses by using the input above the tree. Larger beam sizes show more translation hypothesis, but the tree may become cluttered as the number of nodes of the tree increases, as seen in Figure 4.11. Therefore, this setting must be individually chosen depending on sentence length and context. The tree is expanded and shrunk dynamically based on the current beam size, so that context during hypotheses exploration is not lost when increasing or decreasing the current beam size.

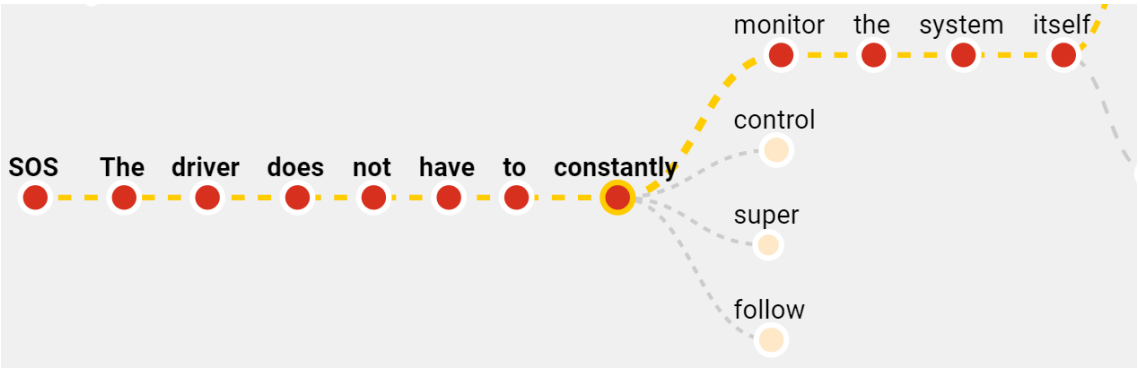

Figure 4.12: After selecting the node "constantly", the tree expands, showing three alternative candidate words "control", "super" and "follow", that may be selected to continue the translation instead of the current next word "monitor".

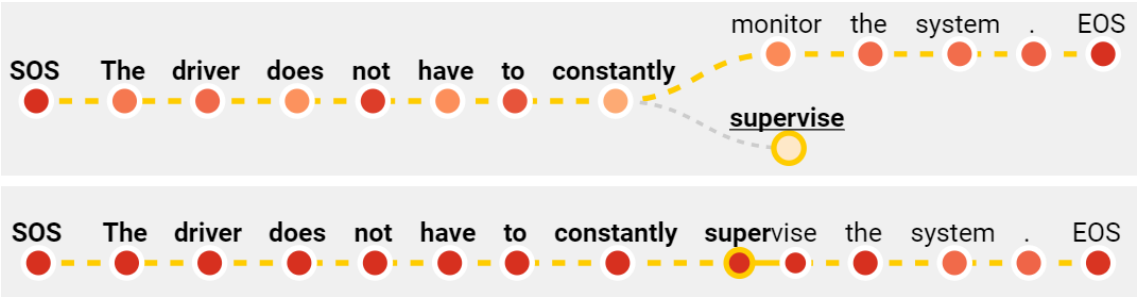

Figure 4.13: A custom correction can be applied at any time during exploration by typing words or phrases. On top, child node "supervise" is appended to the currently selected node "constantly", containing the text typed by the user. On the bottom, selecting the child node updates the Beam Search View, and new translation hypotheses are extended from the current node, continuing the translation as corrected by the user.

4.6.4 Interaction Concept

In order to facilitate interactive exploration and correction of translation hypotheses, users can interact with the Beam Search View in a variety of ways. At the most basic level, interactions can be grouped into two types: (1) navigation and (2) editing. Navigation refers to actions for moving through translation hypotheses, while editing refers to actions that change translation hypotheses in some way. Additionally, the Beam Search View supports two input modalities for interaction: (1) mouse-based input and (2) keyboard-based input. Mouse-based input allows free exploration and adjustment, while keyboard-based input allows rapid post-editing for increased efficiency.

Navigation

Navigation allows the user to explore translation hypotheses. Figure 4.14 shows how the navigation concept is realized for different movements. Navigation using *mouse mode* is afforded through panning inside the view by clicking and dragging. Panning is useful for exploring long translation hypotheses, as the width of the tree may not fit into the current viewport. Additionally, the view can

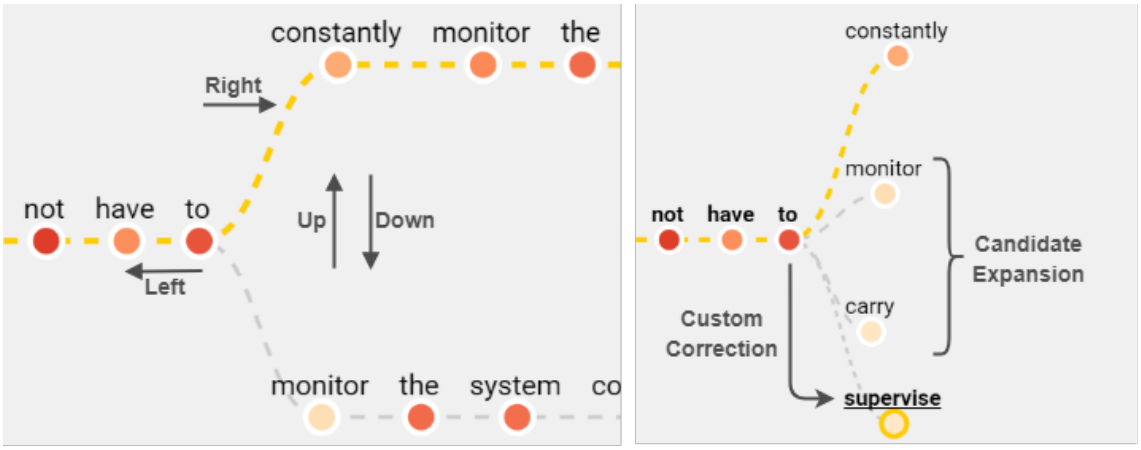

Figure 4.14: The two types of interactive actions in the Beam Search View, navigation (left) and editing (right). **Navigation:** Going left removes the last word from the current translation, going right adds the next word to the current translation. Going up or down cycles through child nodes to decide which node will be selected going right from the current node. **Editing:** Candidate expansion shows alternative words to continue the current translation. Custom correction enables arbitrary user input to continue the current translation.

be zoomed in and out by scrolling the mouse wheel, which can give a better overview for large trees than the standard zoom level. Zooming can also be done by clicking dedicated icons in the upper right corner. A disadvantage of mouse mode is efficiency, as mouse actions are likely slower and possibly imprecise compared to keyboard inputs. However, mouse mode is the more intuitive way to navigate through the Beam Search View, as standard interaction techniques such as panning and zooming are widely used, as for example in online mapping applications.

Keyboard mode enables fast navigation through the arrow keys. Pressing the left arrow key [←] moves the current word back to the parent node. This allows the user to back-up in case words were mistakenly added to the corrected translation. Pressing the right arrow key [→] moves the current word to a child node, thus extending the current translation hypothesis by a word. By default, the first child node is chosen, as child nodes are sorted based on overall the overall probability of translation hypotheses from top to bottom. Pressing the up [↑] and down arrow key [↓], the user can cycle through the child nodes of the current word in either direction to select which one he wants to move to at the next step. In addition, [TAB] and [BACKSPACE] can be used which act analogously to the left and right arrow keys, respectively. The use of [BACKSPACE] for removing the current word from the hypothesis and [TAB] for moving forward is familiar from text processors, where they are used to delete or move characters, respectively. Because these actions are difficult to discover for users naturally, a help text is displayed in the lower left corner.

Editing

Editing encompasses all actions related to choosing, correcting or selecting a translation hypothesis. *Selecting* refers to the act of choosing a (partial) translation hypothesis as the correct translation. By selecting any node, the current translation is set to the prefix defined by the path starting from the selected node and its ancestors up to the SOS node. Selecting an EOS node sets a complete hypothesis as the correct translation.

If the user is not satisfied with any child node of the current node to extend the hypothesis, a *candidate expansion* can be performed. Figure 4.12 shows the effect of expanding the node "constantly" in a sample translation. Since the user rejects "monitor" to continue the partial translation, three child nodes "control" and "super" (likely as subword for "supervise") and "follow" are appended. The user can then select either candidate word to extend the translation. This will cause the beam search to rerun with a lexical prefix constraint and update the tree accordingly. By allowing arbitrary text input at any point, the Beam Search View effectively enables standard post-editing, while also benefiting from the NMT model. As the NMT model continues the translation starting from the last custom correction, the user does not have to correct the entire translation, instead only needs to focus on editing a single part at a time.

In case none of the candidate words are suitable to extend the current partial translation, the user can start a *Custom Correction*, which allowing them to input arbitrary words by typing them at any point during editing or navigation, as seen in Figure 4.13. As soon as the user types, a child node is appended to the current node containing the typed text. In order to differentiate custom edits from other nodes, the text is underlined. Selecting a custom edited node behaves the same as selecting any candidate node. In mouse mode, a translation hypothesis is selected by clicking on an EOS node. For any other node, clicking selects the related prefix translations and also causes a candidate expansion of the current node. In keyboard mode, selection is performed implicitly by navigating to a node as described before. Candidate expansion is performed by pressing the ENTER key.

4.7 Attention View

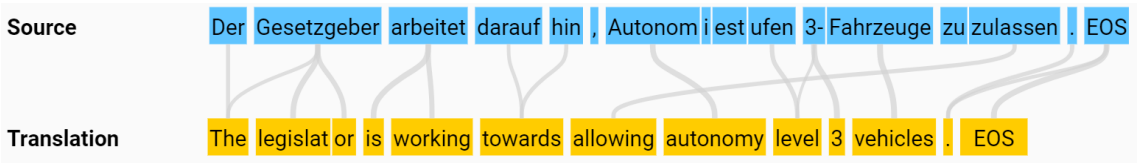

Figure 4.15: The Attention View visualizes the attention values of the current translation as a weighted graph. Each word is represented as a node, and an attention weight between a source and target word is represented by a link. The magnitude of attention weights is mapped linearly to line thickness, and attention weights under a threshold are hidden to minimize visual clutter. Hovering over a word will highlight connected words in the other sentence and the associated attention lines.

4.7.1 Description

The *Attention View*, related to task **T3**, visualizes the attention weights of the model for a particular source and target sentence pair as a weighted graph, based on Strobelt et al. [SGB+18]. Source and target words are linked, with the width of the link mapped to the magnitude of the related attention weight of these words, as seen in Figure 4.15. Hovering over a source token results in the target tokens being highlighted according to the attention values of the source token, allowing the user to see which target tokens were generated due to the hovered token at a glance. Inversely, hovering over a target token highlights the relevant source tokens, again based on the attention values.

Some related works visualized attention weights as a matrix [BCB14]. Using a matrix can be useful for static visualizations or images, where space is no concern. However, a matrix visualization has the disadvantage of creating a lot of whitespace for sparse matrices because the area of the matrix grows quadratically with the length of the sentences, while the number of non-zero entries grows linearly. Attention matrices are usually sparse, as typically each target word has high attention for only a few source words at most. Another disadvantage is that the target sentence must be displayed vertically in a matrix visualization, which goes against the typical reading flow from left to right. Therefore, a compact node-link visualization was chosen to decrease whitespace, leaving more space for the remaining views.

4.7.2 Usage Scenarios

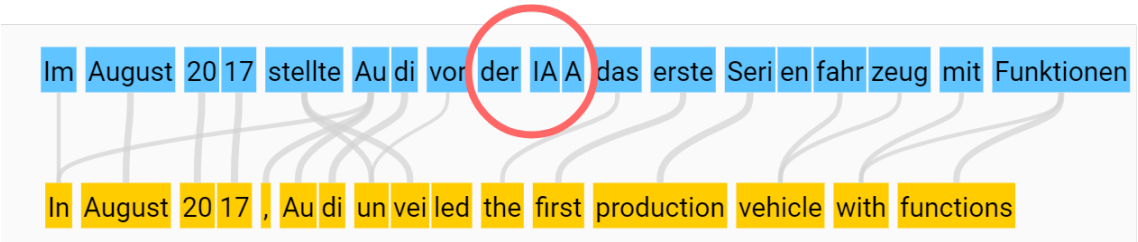

Figure 4.16: Attention View for a sample translation showing under-translation by the NMT model. Note how the phrase “der IAA” was not translated, which can be seen by the missing attention weights coming from these words.

While the Attention View is interesting to gain insight into the behaviour of the NMT model, it is also useful for analyzing a translation hypothesis. By visualizing links between source and target words, irregularities pointing to over- or under-translation can become more obvious, helping the user to find such errors. In Figure 4.16, attention weights for a sample sentence are shown, indicating an under-translation made by the NMT model. Note how it is visible by looking at the missing attention weights that the NMT model skipped translating the source phrase “der IAA”, instead of continuing with the phrase “das erste Serienfahrzeug”. Also, note that despite under-translation, the translation is still fluid, so that simply reading the translation may not reveal this translation error. Therefore, the user must keep the source sentence in mind while checking the translation hypothesis, which is helped by the Attention View as seen by the example.

Over-translation is a well-known problem of NMT models [KK17]. Figure 4.17 shows the Attention View for a source and translation sentence showcasing over-translation. Here, the German source word “jedoch” is translated twice by the NMT model. Note, that the model translates the source

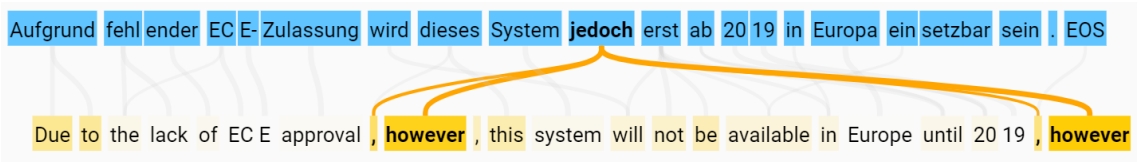

Figure 4.17: Attention View for a sample translation showing over-translation by the NMT model. The NMT model correctly translates source word "jedoch" as "however" but incorrectly places it at two positions that make sense for this word, as seen by two prominent attention weights going into "jedoch".

word correctly, and that both occurrences of "however" in the translation are placed at sensible positions, which might cause a reader to overlook this translation error. The Attention View makes over-translation apparent by visualizing attention weights of multiple target words going into a single source word, thus aiding the user in identifying such translation errors.

4.8 Domain Adaption

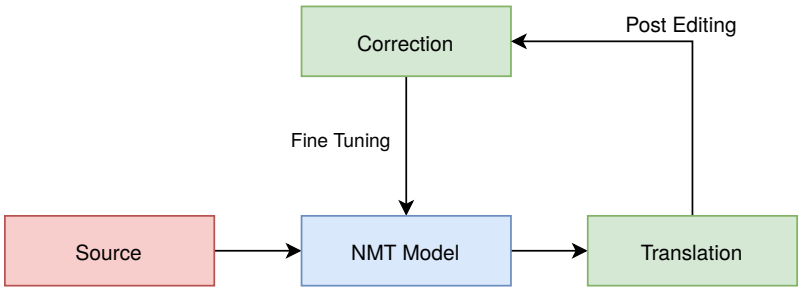

Figure 4.18: Domain Adaption workflow of NMTVis. The NMT model translates a source sentence, producing a translation. The user post-edits the translation using the system, resulting in a correction. Finally, the NMT model is fine-tuned with the correction in an online fashion.

An essential functionality of NMTVis is its ability to incorporate corrections of users into the system, allowing the model to adapt to the specific domain of the current document. Because the NMT model is pre-trained on general data, this feedback can help the model to learn domain-specific terminology or semantic differences between the general corpus and the domain-specific document. Figure 4.18 shows the domain adaption workflow of NMTVIs. After multiple sentences were corrected by the user, he can choose to retrain the current model by selecting the appropriate option in the Document View. The model is then fine-tuned using the corrected sentences. As described in Section 3.5, model centric fine-tuning is performed by freezing all model weights except those of the output layer of the decoder. Afterward, the system translates the uncorrected sentences again, possibly improving translation quality by learning from the corrected sentences.

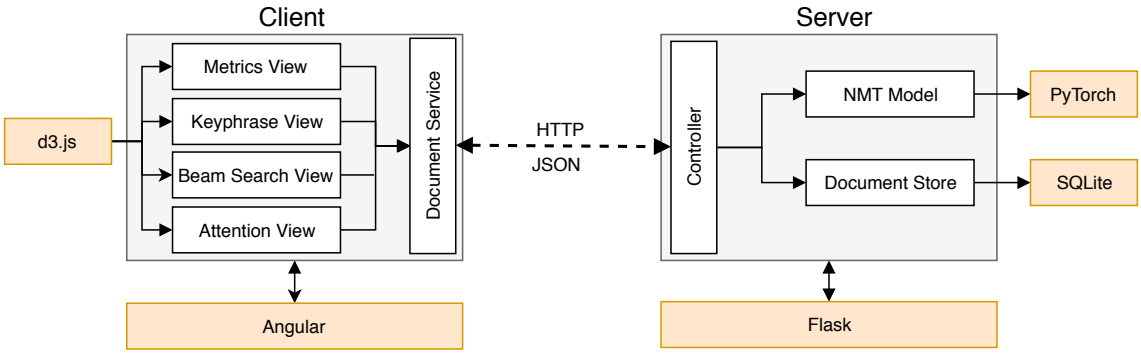

Figure 4.19: The client-server architecture of NMTVis. Orange boxes represent external dependencies, white boxes represent major system components.

4.9 Implementation

The system was designed as a client-server application running in a web browser. The NMT model was implemented in PyTorch[1]. Tokenization of documents was performed using spacy[2]. The client and server exchange data through a common REST-like HTTP API. All visualizations were made with d3.js[3], which provides a low-level API for creating interactive visualizations using Scalable Vector Graphics (SVG). For the tree layout in the Beam Search View, the implementation of the Reingold-Tilford algorithm [RT81] in d3.js was used. The client is a Single Page Application (SPA) running on the Angular[4] framework. The backend was implemented with Python 3 and the Flask[5] framework.

Figure 4.19 shows the system architecture including the most important components. On the client side, the visualization views are implemented as separate components that use d3.js, accessing the Document Service to retrieve relevant data from the server. The Document Service handles communication with the server over HTTP, transporting the data in JSON format. On the server side, the Controller handles incoming requests from the client, parses the JSON data and retrieves the data from the Document Store for translated documents, or passes data to the NMT model for translation. The Document Store stores the data using a lightweight SQLite database. The NMT model will be explained in more detail in the following sections.

4.10 NMT Model

As part of the system, an NMT model translating from German to English was developed and trained based on the theoretical foundations as explained in Chapter 2. A standard attentional encoder-decoder architecture as introduced by [LPM15] was chosen and implemented in PyTorch. Figure 4.20 shows the architecture of the model including layers, connections and layer sizes. The

[1] https://pytorch.org/
[2] https://spacy.io/
[3] https://d3js.org/
[4] https://angular.io/
[5] http://flask.pocoo.org/

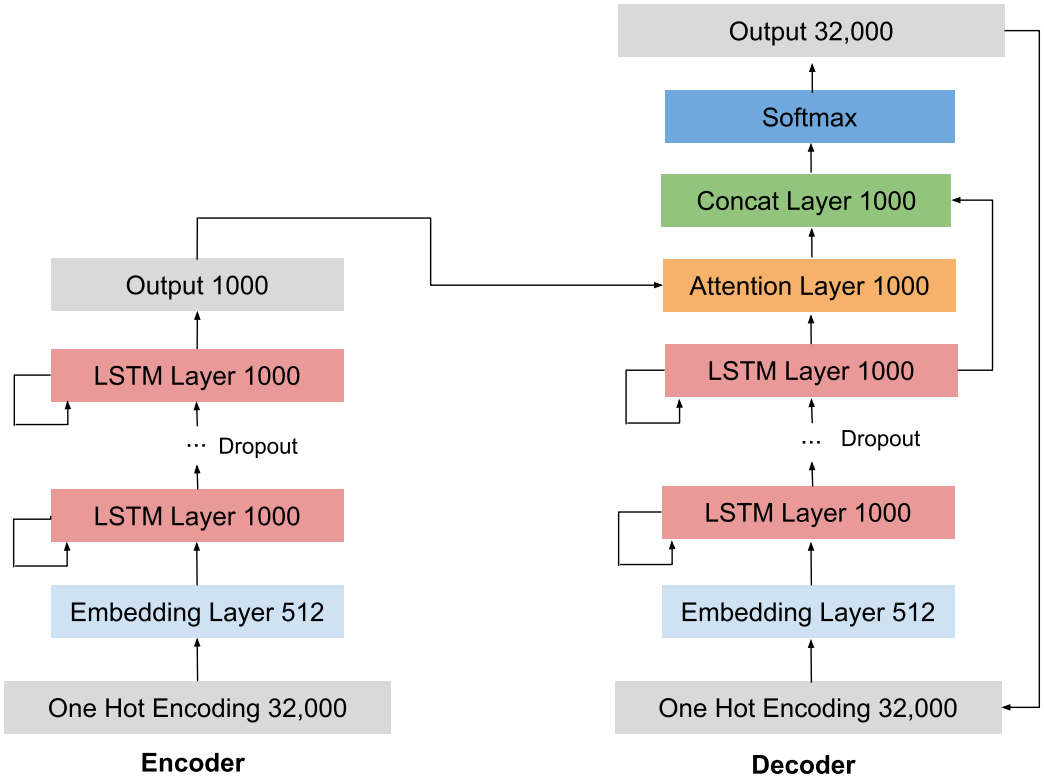

Figure 4.20: The attentional encoder-decoder architecture implemented in this thesis. Each box represents a NN layer, and the numbers signify the layer size.

encoder uses an embedding size of 512, with two stacked, bi-directional LSTM layers with 1000 hidden units each. The decoder also uses an embedding size of 512, and two stacked LSTM layers with 1000 hidden units. For regularization, the dropout mechanism with rate 0.1 is used for the weights between the LSTM layers. The *general* attention mechanism as described in Equation 3.2 was chosen, in combination with the input feeding approach also introduced by Luong et al. [LPM15]. Subword units [SHB15] are used with a shared vocabulary of size 32,000 to handle unknown words and to keep the size of the vocabulary reasonably small. For beam search decoding, a beam size of 3 is used, which was shown to bring sufficient benefit [WSC+16].

4.10.1 Data Sets

The choice of the data set for training an NMT model is crucial to ensure adequate translation quality. Because NMTVis does not focus on a specific domain, a general data set had to be used for training. At first, the IWSLT'14 [CNS+14] DE-EN data set was chosen, containing over 160,000 English and German sentences adapted from TED talks. This relatively small data set allows for fast training time and smaller vocabulary, at the expense of translation quality and model generalization. After initial testing, the model trained on this data was deemed insufficient regarding translation quality due to limited vocabularies resulting in many unknown words during translation.

	WMT'16			IWSLT'14		
	Train	**Dev**	**Test**	**Train**	**Dev**	**Test**
Sentences	4.5M	2.2k	3k	160k	7.3k	6.8k
Total Words DE	107k	44k	62k	3.1M	142k	126k
Total Words EN	114M	47k	65k	3.3M	150k	131k
Distinct Words DE	2M	10k	13k	114k	17.9k	14k
Distinct Words EN	971k	8.1k	9.9k	53k	12.5k	9.4k

Table 4.1: Number of sentences, words, and distinct words (case-sensitive, including punctuation) of WMT'16 and IWSLT'14 DE-EN data sets used for development, training and evaluation. Note that these statistics were computed on the tokenized data sets, before BPE was applied. Numbers are rounded to thousands (k) and millions (M) for readability.

NMT models exhibit a steep learning curve in relation to the size of training data Koehn and Knowles [KK17], which means that translation quality starts low, and then rises quickly with increasing training data size. Many large public data sets are available, drawing from a variety of sources such as news articles, subtitles or Wikipedia articles, where the existence of a document in multiple languages is needed. For development, training and evaluation, the DE-EN data set from the 2016 ACL Conference on Machine Translation (WMT'16) [BCF+16] shared news translation task was chosen, containing over 4 million sentences. This popular data set was used in several related works on NMT (e.g. [DN17], [SHB16]) and allows for easy comparison of translation quality. We use newstest2016 as our test set and newstest2015 as the development set.

Table 4.1 shows the number of sentences and (distinct) words of the translation data sets used in this thesis. The WMT'16 training set has over 100 million words in total for each language, compared to 3 million for IWSLT'14. It is also notable that English data sets have a consistently higher total word count across all types of sets compared to German but also contain significantly fewer distinct words, e.g. roughly $100,000$ for German and $50,000$ for English in the IWSLT'14 training set. An explanation might be morphological differences between both languages, e.g. conjugation of verbs in German compared to English.

4.10.2 Training

All training was carried out on an NVIDIA GeForce GTX Titan X GPU with 12 GB RAM using Stochastic Gradient Descent and Adam [KB14] with an initial learning rate of 10^{-4}, and using the cross entropy loss function. All hyperparameters were determined using grid search on the development set. A batch size of 256 was used, training for a total of 20 epochs. The training corpus was reshuffled after each epoch. Gradient clipping was used to prevent the exploding gradient issue, with a maximal gradient norm of 5. For further regularization, weight decay with a value of 10^{-5} was used. For model selection, the model with the lowest BLEU score on the test set during training was used as the best model.

Figure 4.21 shows the average cross entropy loss per sentence on the training set and the validation data for IWSLT'14 and WMT'16 data sets. Note that for IWSLT'14, the validation loss is lower at first due to the use of dropout, which is disabled when evaluating the model on the validation set,

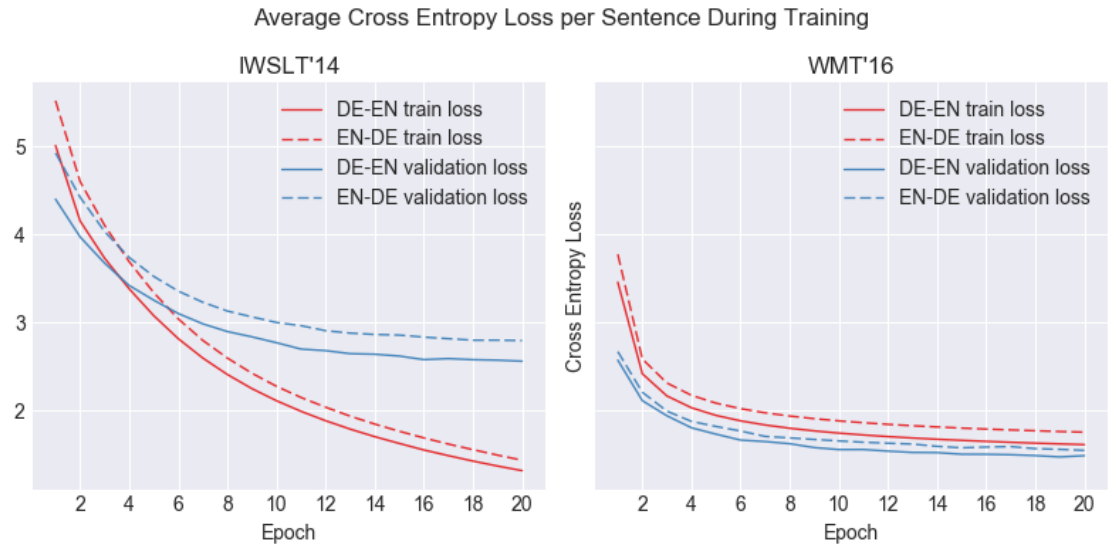

Figure 4.21: Average cross entropy loss per sentence of NMT model trained on IWSLT'14 and WMT'16 DE-EN data on training set and validation set for DE-EN and EN-DE translation directions.

leading to a high initial loss for the training data. The validation loss stops decreasing roughly at epoch 14, while the training loss keeps steadily decreasing. This means that the model is most likely starting to overfit on the training data, making further training useless. The loss progression for WMT'16 has major differences compared to IWSLT'14. First of all, both training and validation loss start off much lower at about 3.5 and 2.5 respectively compared to IWSLT'14. Both losses quickly descend at almost equal rates, with validation loss stabilizing around 1.5 after 10 epochs. The training loss also never goes below the validation loss and does not seem to decrease more rapidly in comparison, which points to the fact that the NMT model does not overfit as much as for IWSLT'14.

The most likely reason for these differences in performance and loss of the data sets lies in the large size difference between them. The training set for IWSLT'14 only contains $160,215$ sentences, compared to $4,500,966$ for WMT'16. This means that the model was trained on $20 \cdot 160,215 = 3,204,300$ sentences for IWSLT'14, less sentences than for a single epoch on the WMT'16 set. On the other hand, training time for IWSLT'14 is significantly shorter, with 9.5 hours for German $\rightarrow$ English (DE-EN) and 14 hours for English $\rightarrow$ German (EN-DE) compared to 5 days and 6 days for WMT'16, respectively. The small size of IWSLT'14 may also explain the issue of overfitting, as the model is likely too complex with too many weights for this data set while being a good fit for the larger WMT'16 data set. Looking at the difference in loss regarding translation direction, we see that the loss for EN-DE is consistently higher than for DE-EN for both data sets. This difference is notably higher for IWSLT'14 compared to WMT'16, where EN-DE loss is only slightly higher. Looking at the loss curves in general, DE-EN loss seems to decrease at the same rate as EN-DE loss, suggesting that the model does not converge faster for one language direction, but that translating EN-DE is harder for the NMT model to learn.

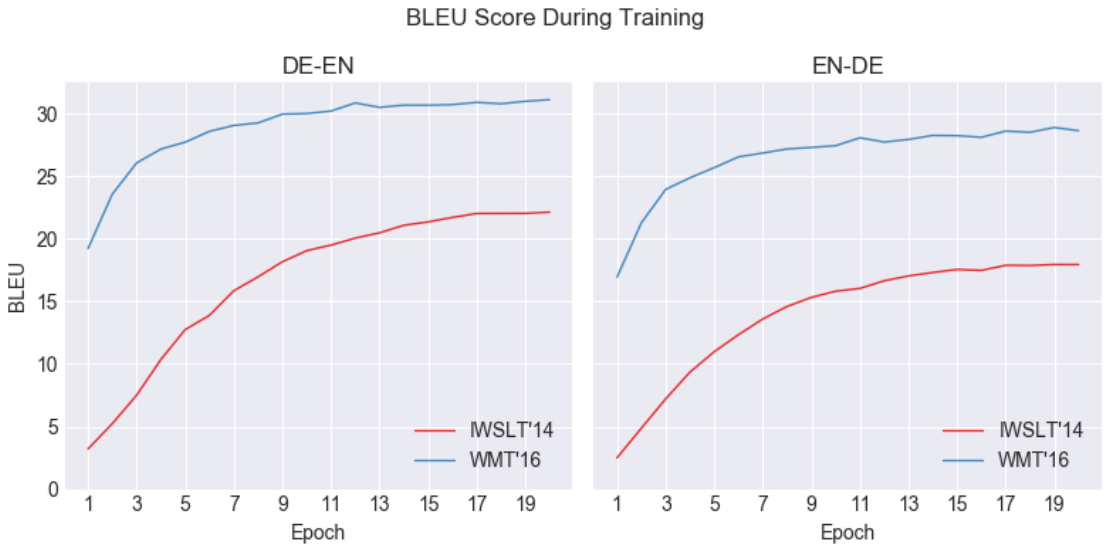

Figure 4.22: The BLEU score computed on the validation set for each epoch during training for IWSLT'14 and WMT'16, and translation directions DE-EN and EN-DE.

While tracking the average loss makes sense to control overfitting and generalization, it does not explain how well the model actually performs on translation tasks. Therefore, the BLEU score was also computed on the validation set of each data set after each epoch, as seen in Figure 4.22. For WMT'16, the first epochs lead to a large increase of the score, with a slow incline up to epoch 12, after which the score remains mostly stable around 30, ending after 20 epochs at 31. For IWSLT'14, the score starts at around 3, increasing linearly for several epochs, reaching a plateau after 15 epochs at around 20. The NMT model trained on WMT'16 performs significantly better with a BLEU score of 31 compared to a score of 20 for IWSLT'14. The model also generalizes better as seen by evaluating training and validation loss. The main disadvantage of this model is the long training time due to the size of the data set, which has no impact during translation, however. Therefore, the WMT'16 model was chosen for the final system.

4.10.3 Model Evaluation

Table 4.2 shows the final results of NMT models on the WMT'16 and IWSLT'14 data sets for German-English translation in both directions (EN-DE and DE-EN). We report the BLEU on the untokenized test sets as specified in Section 4.10.1, using the best model selected by the highest BLEU score on the validation set and training time for each model. The evaluation on the test sets confirm the preliminary results during training. The model trained on WMT'16 achieves significantly higher BLEU for both German → English (31.1 vs. 21.1) and English → German (28.6 vs. 17.9). Looking at translation direction, we see that the model performs better at translating into English than into German, with a difference of +2.5 for WMT'16 and +3.2 for IWSLT'14. This result, which matches with findings by Sennrich et al. [SHB15], suggests that learning to generate German sentences is harder for NMT models, which may in part be due to more complex grammar.

	WMT'16		IWSLT'14	
	DE-EN	**EN-DE**	**DE-EN**	**EN-DE**
NMTVis	31.1	28.6	21.1	17.9
Luong et al. [LPM15]	24.9	25.9	-	-
Freitag et al. [FWP+14]	-	-	25.8	23.3
Denkowski and Neubig [DN17]	33.5	-	-	-
Sennrich et al. [SHB16]	38.6	34.2	-	-

Table 4.2: Comparison of Bleu scores of our approach NMTVis and related NMT systems for German-English (DE-EN) and English-German (EN-DE) translation on the WMT'16 and IWSLT'14 test sets.

We also compare our model against multiple other NMT systems that were evaluated on WMT and IWSLT data sets. Freitag et al. [FWP+14] achieve significantly higher scores on IWSLT'14 for both directions, compared to our model, with a difference of +4.8 for DE-EN and +5.4 for EN-DE. For WMT'16 our model performs better than Luong et al. [LPM15], likely due to the use of subword-units with BPE. For DE-EN, Denkowski and Neubig [DN17] performs slightly better than our model, with an improvement of +2.4 points. The system of Sennrich et al. [SHB16] achieves the greatest Bleu score of 38.6 for DE-EN and 34.2 for EN-DE, using ensemble models and synthetic parallel data sets generated by backtranslating from the target language. These results show that our base model for WMT'16 has comparable performance to related systems, making it adequate for our use case, but that there is still a lot of room for improvement using advanced techniques such as ensembling and synthetic data sets.

5 User Study

Evaluating a novel, interactive VA systems is a challenging task, due to the ambiguous, high-level goals of these systems and their dependency on human users [KKE10]. User studies are a common way of evaluating VA systems. This chapter will describe the web-based user study that was conducted as part of this thesis.

5.1 Goals

The goal of the user study was to evaluate the following aspects of NMTVis:

Effectiveness of the system regarding its main tasks as defined in Section 4.1, especially sentence selection of erroneous translations, exploration, and analysis of translation hypotheses, and post-editing of machine-generated translation.

Ease of Understanding of the views and visualizations. The visual encoding should be intuitive and easy to interpret, and present useful information to users.

Ease of Interaction with the views and visualizations to enable effective and efficient execution of the tasks.

For evaluating effectiveness, an experimental study could have been conducted, e.g. by measuring the time needed for post-editing and comparing against a baseline system. Due to time constraints and the high effort needed for conducting such an experiment, a user study was chosen instead, asking users for their subjective rating regarding the usability and effectiveness of the system through a questionnaire.

5.2 Study Design

We chose to conduct a web-based user study to evaluate NMTVis, which is afforded by fact that the system was implemented as a web application, including user management features. There are a multitude of advantages and disadvantages of web-based user studies and experiments [Rei00]. One advantage of such an unsupervised approach is that participants cannot be influenced by the presence of a researcher, and are free to explore the system as they desire, removing experimenter bias. This might result in more credible feedback since participants might feel pressure to give positive feedback in a supervised setting. Web-based studies also reduce cost and time, as no special equipment, lab space or inventory is needed, enabling to scale the study for a large number of participants. However, there are also disadvantages of such an approach. Technical variance is one disadvantage, as users access the web-based system on a multitude of operating system, browsers, and platforms. Therefore, experimental conditions can differ between users, and issues can arise

due to technical problems. Participants may drop-out during the study unexpectedly due to a lack of accountability. Additionally, there is a lack of experimental control, and participants may not take questions or ratings seriously. In order to minimize these disadvantages, special care was taken to test the system for different technical configurations and environments. An introduction was written that explains the goals and tasks of NMTVis, as well as the purpose of the user study in simple language to minimize drop-out due to confusion or information overload.

For the user study, each participant was sent an email containing a web link to the system, that explained the details of the user study. Participants then could freely use the system for an arbitrary amount of time. By pressing a "Finish Study" button, a participant could exit the application before being redirected to a questionnaire. Participants were asked to perform the following three tasks:

1. **Sentence Selection:** Using the Metrics and Keyphrase View, sentences containing erroneous translations should be found.
2. **Exploration of Translation Hypotheses:** Using the Beam Search and Attention View, translation hypotheses should be explored.
3. **Correction of Translation Hypotheses:** Using the Beam Search View, sentences found in the first task should be corrected.

No further instructions were given, and participants were free to perform these tasks in any order for any length of time. After performing all tasks, participants were prompted to answer the questionnaire, which is shown in Appendix A. We used a 7-point Likert scale for all rating questions, as is done by several standard usability questionnaires [Dav89; Lun01]. We adapted a question from the After-Scenario Questionnaire [Lew95] to measure overall ease of task completion. The questions are divided into three types: (1) questions regarding the effectiveness of the system, (2) questions regarding the intuitiveness of visualizations and interaction (3) open questions for free-form feedback. Specific questions were grouped for each of the four evaluated views (Metrics View, Keyphrase View, Beam Search View, and Attention View). For general questions about the effectiveness, we also asked if the participant would prefer to use NMTVis over a traditional text-based system for (1) small documents with less than 20 sentences and (2) large documents with more than 100 sentences.

5.3 Data Set

In order to ensure comparable study settings, a pre-selected document was chosen and uploaded to the system for each participant. Because we cannot assume specific domain knowledge, e.g. knowledge of medical or legal vernacular from the participants, a document had to be chosen that could be translated with common knowledge. At the same time, domain-specific words that do not appear in the out-of-domain training data should still be included to test the effectiveness of the system for domain-specific documents. Therefore, the the German Wikipedia article about autonomous driving [1] was chosen for the user study. The chosen document contains 92 sentences, with a maximum length of 48 words and minimum length of 2 words. The document contains 1822 words in total and 843 distinct words (case-sensitive, including punctuation), with an average

[1] https://de.wikipedia.org/wiki/Autonomes_Fahren

Unter autonomem Fahren (manchmal auch automatisches Fahren, automatisiertes Fahren oder pilotiertes Fahren genannt) ist die Fortbewegung von Fahrzeugen, mobilen Robotern und fahrerlosen Transportsystemen zu verstehen, die sich weitgehend autonom verhalten.
Autonomiestufe 3: Bedingungsautomatisierung.
Voraussetzung ist das Vorhandensein von Sensoren (Radar, Video, Laser) und Aktoren (in der Motorsteuerung, der Lenkung, den Bremsen) im Fahrzeug.
Am 25. Januar 2017 hatte die Bundesregierung einen Gesetzentwurf beschlossen, der autonomes Fahren auf den Straßen des Landes unter bestimmten Voraussetzungen zulassen soll.

Table 5.1: Four sample sentences from the German Wikipedia article on autonomous driving.

Keyphrase	**Frequency**
Autonomiestufe	8
Verkehrsgerichtstag	3
Automatisierungsstufe	2
Spurhalten	2
Dilemmasituationen	2
Längsführung	2
Tempomat	2
Mikroprozessorsysteme	1
Sensordaten	1
Spurhalten	1
SAE-Level	1

Table 5.2: Top 10 keyphrases extracted from the German Wikipedia article on autonomous driving used for the user study.

of 19.8 words per sentence. Table 5.1 shows a manually chosen sample of sentences from the document that exhibits most of its relevant characteristics. The first sentence is of moderate length and complexity, containing 30 words and an interjection in parentheses, as well as domain-specific words such as “automatisiertes Fahren”. The second sentence is comparably short but contains the compound word “Bedienungsautomatisierung”. The third sentence contains more technical jargon (“Radar”, “Laser”, “Aktoren”) and the last sentence contains legal terms (“Gesetzentwurf”, “Bundesregierung”). Table 5.2 shows the top 10 domain-specific keyphrases and their frequency. As we can see, the keyphrases reflect the domain of the document at hand, with multiple keyphrases directly related to autonomous driving such as “Autonomiestufe”, “Automatisierungsstufe” and “SAE-Level” and technical jargon such as “Längsführung” and “Mikroprozessorsysteme”. These sample sentences and keyphrases validate the use of this document for the user study.

	Participant Background Knowledge		
	Median	**Mean**	**Std. Dev.**
German Language Proficiency	7	6.4	1.6
English Language Proficiency	5	4.9	0.8
Visualization Knowledge	4	3.9	1.6
Machine Translation Knowledge	2	2.8	2

Table 5.3: Subjective background knowledge ratings of participants including language proficiency, visualizations and machine translation knowledge. Each question was asked on a 7-point Likert scale.

5.4 Participants

Table 5.3 shows the results for the subjective ratings of background knowledge for German and English language proficiency, as well as visualization and machine translation knowledge. There were 15 participants that took part in the user study. Five of the participants were researchers and students at the Institute of Natural Language Processing (IMS) of the University of Stuttgart. Six of the participants were researchers at the Institute for Visualisation and Interactive Systems (VIS) of the University of Stuttgart. From the remaining participants, two were Software Engineers. 12 participants were native German speakers, while two participants rated their German language proficiency as good (subjective ratings of 5 and 6), and one participant rated their German skills as poor (subjective rating of 1). Regarding English language proficiency, subjective ratings were high (median = 5, mean = 4.9), however, no native speakers were among the participants. There was some prior visualization knowledge (median = 4, mean = 3.9) due to the VIS participants. Knowledge about machine translation was low across all participants (median = 2, mean = 2.8), with one outlier with a subjective rating of 7 and the largest standard deviation of 2 among all background questions.

5.5 Results

This section will present the results of the questionnaire regarding the effectiveness, ease of understanding, and ease of interaction of NMTVis.

5.5.1 General Effectiveness

The results for the general effectiveness of NMTVis are shown in Table 5.4. Overall, ease of task completion was rated highly (median = 5, mean = 5.1), with a single outlier with rating 1. Two participants gave the highest positive subjective ranking of 7 regarding ease of task completion. There is a clear distinction regarding preference of the system for post-editing of small and large documents compared to a simple text-based editing approach. Preference for using NMTVis for large documents (median = 6, mean = 5.8) is higher compared to small documents (median = 4,

	General Effectiveness		
	Median	**Mean**	**Std. Dev.**
Ease of Task Completion	5	5.1	1.5
System Preference for Small Documents	4	4.5	1.5
System Preference for Large Documents	6	5.8	0.9

Table 5.4: Ratings for subjective effectiveness of each evaluated view on a 7-point Likert scale.

mean = 4.5). Additionally, the standard deviation is smaller for large documents (0.9) compared to small documents (1.5). On average, participants rated their preference for large documents higher by 1.2 points.

5.5.2 Effectiveness

	Effectiveness		
	Median	**Mean**	**Std. Dev.**
Metrics View	6	5.9	1
Keyphrase View	5	4.5	1.5
Beam Search View	6	5.8	1.4
Attention View	5	5.5	0.8

Table 5.5: Ratings for subjective effectiveness of each evaluated view on a 7-point Likert scale.

The results of the user study regarding the effectiveness of the different views for different tasks are shown in Table 5.5. Overall, the effectiveness of the views was rated highly. In particular, the Metrics View and the Beam Search View received high ratings with median ratings of 6, and mean ratings of 5.9 and 5.8 respectively. Close behind, the Attention View received a median rating of 5 and a mean rating of 5.5, with a standard deviation of 0.8, which is the lowest among all views. The Keyphrase View received the lowest ratings regarding effectiveness for finding erroneous translations, with a mean rating of only 4.5, and median rating of 5, as well as the highest standard deviation of 1.5. Two participants rated the effectiveness of the Keyphrase View for finding erroneous translations with 2, and three participants rated it with 7, showing that subjective ratings differ significantly regarding its effectiveness.

5.5.3 Visualization

The results of the user study regarding ease of understanding and intuitiveness of visualizations are shown in Table 5.6. Generally, ease of understanding and intuitiveness regarding the visualizations was rated highly. The Metrics View, Keyphrase View, and Beam Search View all received high ratings with medians of 7, and means of 6.6, 6.5 and 6.2 respectively. In particular, the Metrics View achieves the highest mean rating with the lowest standard deviation of 0.6. The Attention

	Visualization		
	Median	**Mean**	**Std. Dev.**
Metrics View	7	6.6	0.6
Keyphrase View	7	6.5	1
Beam Search View	7	6.2	1.2
Attention View	6	6.1	1.1

Table 5.6: Ratings for ease of understanding and interpretability of the visualizations for each evaluated view on a 7-point Likert scale.

View received a median rating of 6, with a mean rating of 6.1, close behind the other views with single outlier with a rating of 3. Compared to all other question groups, ease of understanding of the visualizations received the highest ratings on average.

5.5.4 Interaction

	Interaction		
	Median	**Mean**	**Std. Dev.**
Metrics View	6	6.1	0.9
Keyphrase View	7	6.4	1
Beam Search View	5	5	1.8
Attention View	6	6	0.8

Table 5.7: Ratings for ease of interaction with each evaluated view on a 7-point Likert scale.

The results of the user study regarding ease of interaction with the different views are shown in Table 5.7. In general, all views were rated positively, with mean and median ratings above 4. The Keyphrase View had the highest average rating (median = 7, mean = 6.4). The Metrics View (median = 6, mean = 6.1) and Attention View (median = 6, mean = 6) were rated slightly lower, with similar standard deviations (0.9 and 0.8). Interaction with the Beam Search View received the lowest ratings overall (median = 5, mean = 5) and also has the highest standard deviation with 1.8. The opinion on Beam Search View interaction seems to differ sharply among participants, with two low ratings of 1 and 2, and two high ratings of 7, reflecting in its high standard deviation of 1.8.

5.5.5 Free Text Feedback

In addition to the rating questions, we also solicited free text feedback through open questions from the participants. Regarding functionality and views, a few trends became apparent in the feedback. The Metrics View was mentioned positively for a majority of participants, especially those with a visualization background. Multiple participants noted its usefulness for quickly finding erroneous translations through brushing and linking. For example, one participant stated that the Metrics View "gives nice possibilities to look for different kinds of errors" and another that it "makes it really

easy to find translation errors (mostly confidence & penalty)". Several participants gave positive feedback regarding the suggestions from the Beam Search View. For example, one participant mentioned that the Beam Search View "automatically gives you alternatives which speed up the correction of translation". One participant found that "navigation is mostly intuitive, and it only takes a few minutes to get accustomed with the tool" and that they would find themselves "using it efficiently after short time of training, given a few tweaks". Concerning the Attention View a participant noted that it is "interesting to see the differences in the sentence structure of different languages" and that they "never saw something like this before in an online translation tool", which highlights the use of NMTVis for making the underlying NMT model more interpretable.

Most feedback regarding aspects of the system that the participants viewed negatively revolved around interactivity and specific features. Multiple participants noted that post-editing and exploration of long translation are challenging using the Beam Search View, due to the limited size of the viewport. Two participants found the yellow-red color scale in the Beam Search View hard to interpret, with one participant pointing out that a diverging color scale, e.g. a red-white-blue scale might be helpful to get a sense where the central value is. This participant also mentioned that long sentences might be laborious to correct in the Beam Search View. Another participant found interaction with the Beam Search View extremely difficult, noting a lack of undo functionality, slow responsiveness, as well as the fact that individual words cannot be deleted from the Beam Search View. Two other participants also noted the slow response time for custom corrections in the Beam Search View. Further, the custom correction was found to be troublesome, noting that "it would be much nicer if I could press enter, then edit the custom text in a text field and use the arrow keys as I am accustomed to". This participant also suggested that it should be possible to freeze parts of a translation so that it remains static throughout interaction with the Beam Search View. Regarding the Metrics View, one participant mentioned that filtering through brushing and filtering has the disadvantage that the local context of a sentence is immediately lost, as the original order of sentences is disturbed. This would make the translation of large paragraph difficult, as the context of each sentence is important during translation. Multiple participants could not find out how to remove a brush in the Metrics View. One participant noted that they would "accidentally highlight a different line (sentence)" in the Metrics View, as hovering over a line selects the respective sentence.

Regarding suggestions for new features, one participant noted that it might be useful to show similar sentences given a selected sentence. This might be helpful to find several related sentences that contain similar errors, e.g. when a particular word is mistranslated by the system. Another suggestion was a view that "that preserves sentence order but still provides confidence visualizations", unlike the Metrics View, that only enables filtering or ordering based on confidence. Confidence scores could be displayed directly next to a translation. Another suggestion for a future system was to "combine a regular text field with auto-completion-like translation suggestions", replacing the Beam Search View. Such an approach would be closer to traditional text-based post-editing systems and could still provide benefits of beam search decoding such as suggesting alternative words.

5.6 Discussion

The user study showed that the system in its entirety and the different views were rated positively regarding effectiveness, interpretability of visualizations and ease of interaction. Overall, the system was perceived as useful for translation tasks, with a clear preference to use the system for

translation large documents compared to a traditional text-based approach. Only the effectiveness of the Keyphrase View was rated comparatively low and received most differing ratings. For ease of interaction, the Beam Search View received the most diverging ratings, suggesting that its interaction concept has potential for improvement. Ease of understanding for the visualizations received high ratings in absolute terms and also higher ratings compared to effectiveness and interaction, suggesting the visualizations are mostly intuitive to understand. Most free text feedback reinforced the ratings, with many participants especially liking the sentence selection mechanism using the Metrics View and the suggestions made by the Beam Search View. However, the free-form feedback also revealed weaknesses and opportunities of the current approach of NMTVis. Based on the qualitative feedback from the participants and our own analysis, the following key challenges were identified:

Interaction In order for NMTVis to become a productivity tool for post-editing, more research regarding efficiency, fluency in interaction must be done. In particular, the Beam Search View interaction should be more responsive, allowing standard text editing features such as undo functionality. For the Metrics View, removal of brushes should be made more intuitive, as multiple users had issues removing them. Sentence selection in the Metrics View should be improved such that accidental selection is minimized.

Visual Encoding Different visual encodings for word probability in the Beam Search View should be investigated to improve interpretability. A diverging color scale may be useful to get a better sense of high and low word probabilities. In the Document View, the confidence score of translations should be visualized in some way, e.g. by mapping the confidence score to the background colour of a translation.

Scalability of Visualizations The sentence-level visualizations lead to issues for dealing with long sentences, as their width scales linearly with sentence length. This makes interactive post-editing of long sentences difficult, as (1) only a part of the source sentence is visible in the Attention View at a time necessitating scrolling and (2) the depth of the graph in the Beam Search View makes it difficult to see the entire graph at once, necessitating zooming and panning. Both issues increase the cognitive load on the user, as he has to actively navigate through the visualizations while keeping the context of the entire sentence in mind, making post-editing slower and less efficient. More research needs to be conducted on how to scale these visualizations better in order to mitigate these disadvantages.

Local Context The user study revealed that the local context of a sentence, made up of its surrounding sentences, is important during post-editing. The different views must make the current context of a sentence visible at any time, and the order of sentences should be restorable. In addition, the Beam Search View should also include some form of context during post-editing.

6 Automated Evaluation

Besides the qualitative evaluation in the user study, we also evaluated the system through a series of experiments. Since translation evaluation is both time and cost intensive, we conducted these experiments in an automated manner. The main questions for the automated evaluation were:

- How well do the metrics used in NMTVis correlate with translation quality?
- How useful are the metrics used in NMTVis for finding erroneous and low-quality translations?
- How well does the model used in NMTVis adapt to domain-specific documents?
- How well does the model used in NMTVis learn to translate rare and domain-specific words?

The following sections will explain the experiments we conducted, and discuss the results we obtained.

6.1 Data Set

	Khresmoi Data Set	
	Test	**Dev**
Sentences	1,000	500
Total Words DE	20,825	8,670
Total Words EN	21,425	9,924
Distinct Words DE	5,730	3,620
Distinct Words EN	4,493	3,180

Table 6.1: Number of sentences, words, and distinct words (case-sensitive, including punctuation) of Khresmoi medical EN-DE data sets used for the automated evaluation. Note that these statistics were computed on the tokenized data sets, before BPE was applied.

For the automated evaluation, a domain-specific document was chosen in order to validate the effectiveness of the system for in-domain data given our NMT model trained on out-of-domain data. We use the Khresmoi EN-DE data set [DHH+17] from the WMT'17 shared biomedical translation task [YNN+17] for our experiments. This data set consists of 1500 sentences related to medical trials, containing complex medical jargon. The data is split into a test set with 1000 sentences and a development set with 500 sentences. Table 6.1 shows detailed statistics for both sets. We use the test set for analyzing possible correlations between metrics and translation quality. For the domain adaption experiments, the test set is used for online training due to its larger size, and the development set is used as a hold-out set for evaluation.

6.2 Correlation of Metrics and Translation Quality

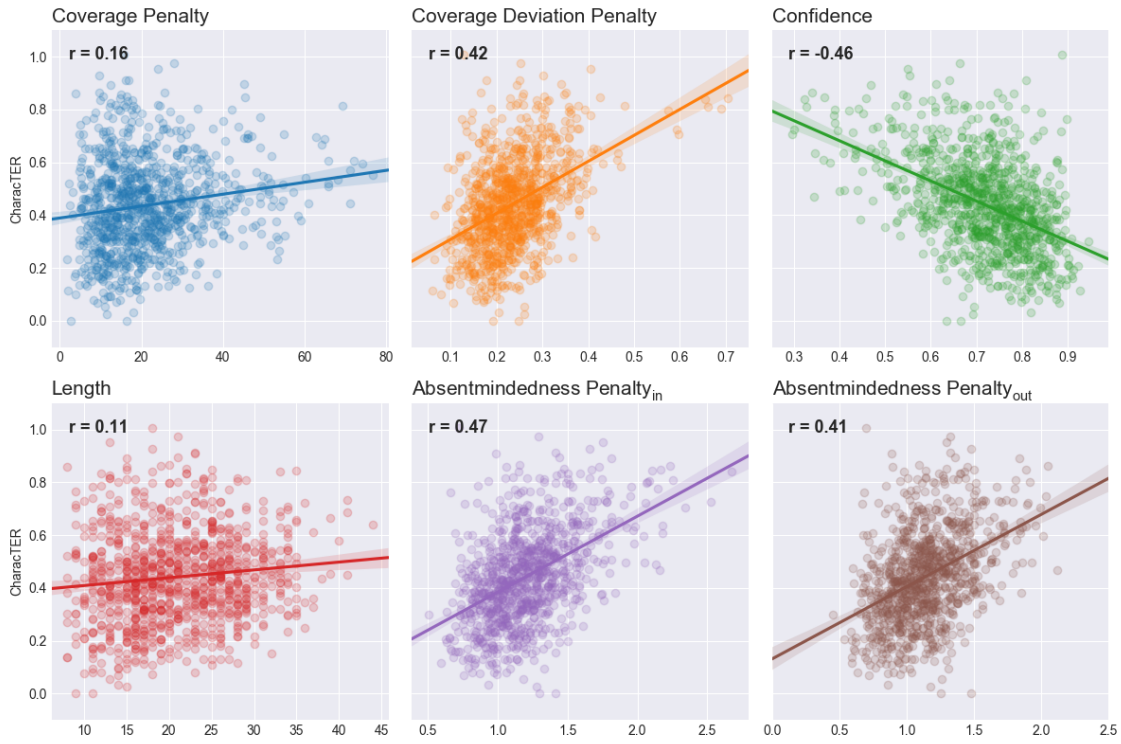

Figure 6.1: Scatterplots for each evaluated metric plotted against CharacTer score on the Khresmoi medical test set. The regression line using the least-squares method is shown, as well as the 95% confidence interval (faded region). Each point represents a sentence. Metric scores are plotted on the x-axis, CharacTer scores are plotted on the y-axis. The Pearson correlation coefficient r is given for each scatterplot.

We introduced the Metrics View (see Section 4.4) which allows selection and filtering of sentences based on certain metrics. In order to evaluate if the chosen metrics are useful to detect sentences with low translation quality, a correlational experiment was conducted. Figure 6.1 shows the scatterplot matrix of each evaluated metric against the CharacTer score (see Section 3.7) for each sentence from the medical Khresmoi test set, as well as regression lines and Pearson correlation coefficient r. The CharacTer score was chosen because a sentence-level metric for translation quality was needed, meaning that Bleu as a document-level metric could not be used. More precisely, for each metric m_i and sentence s_j we calculated the metric score $x_{ij} = m_i(s_j)$, and CharacTer score y_j. The plot for metric m_i then shows the points (x_{ij}, y_j).

The strongest correlations between metric score and CharacTer score is found for AP_{in} with $r = 0.47$. Confidence has the largest negative correlation with $r = -0.46$, with slightly lower magnitude compared to AP_{in} and higher magnitude than AP_{out} ($r = 0.41$). CDP ($r = 0.42$) ranks third overall regarding correlation strength but with a similar appearing distribution as AP_{in} and AP_{out}. CP with $r = 0.16$ has a comparably weak correlation compared to the previous metrics. Notably, sentence length ($r = 0.11$) has the weakest correlation with CharacTer, meaning that the model does not significantly perform worse on longer sentences. One explanation might be that because

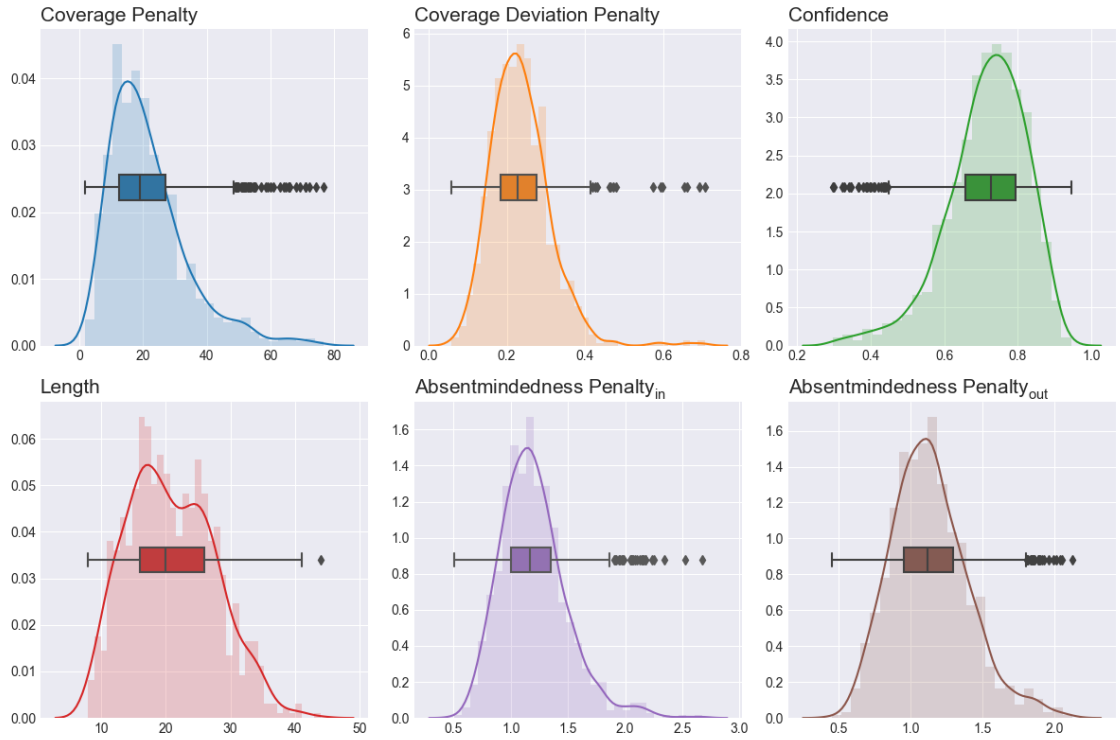

Figure 6.2: The kernel density estimates, histograms and box plots for the scores of each evaluated metric on the Khresmoi medical test set.

the score is normalized by sentence length, the NMT model does not make over-proportionally many mistakes on longer sentences, leading to similar scores for short and long sentences. Looking at the regression lines, we can see a large spread of values around them for all metrics, meaning there is a large variance for CharacTer scores for a given metric score. Most sentence scores also seem to be clustered around the mid values of each metric, as seen in the plot for CDP, where a majority of points are clustered between $x = 0.1$ and $x = 0.3$. At the tail ends, however, we can see clear trends, such as most points with low confidence having high scores or all points with large AP_{in} values having high scores.

We also analyzed the distributions of scores for the different metrics. Figure 6.2 shows the kernel density estimates, box plots and histograms of the distribution for each evaluated metric on the Khresmoi test set. All metrics appear to be roughly normally distributed around the median, except for length which seems to follow a bi-modal distribution. Further, all penalty distributions show a large number of outliers for large scores, creating long tails for their distributions. These outliers may represent erroneous translations of the NMT model. For CP, scores larger than 50 are considered outliers, for CDP larger than 0.4, for AP_{in} larger than 1.9 and for AP_{out} larger than 1.8. Inversely, the confidence distribution has a large number of outliers for low scores (< 0.4), as confidence represents the opposite of a penalty. The box plots show that the middle 50% of scores fall into a narrow range around the median value for all metrics except length.

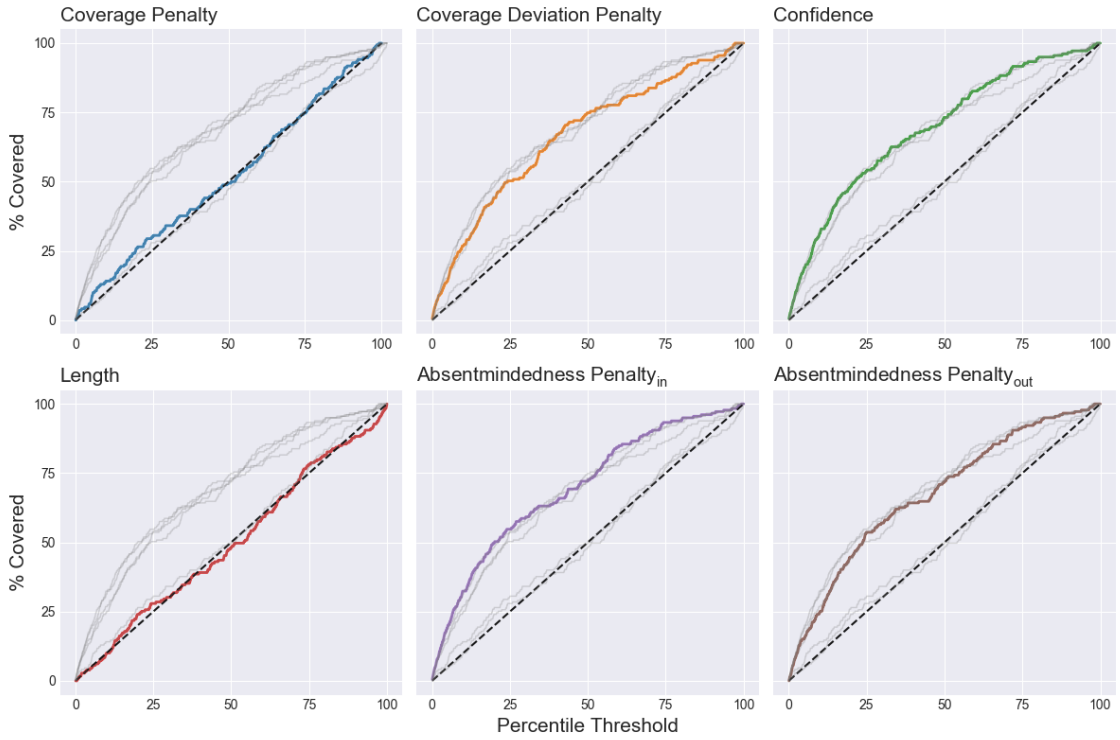

Figure 6.3: Percentage of sentences with low translation quality (CharacTer score larger than median score plus one standard deviation) covered with increasing percentage of sentences considered for each metric on the Khresmoi medical test set. For each metric, all sentences were first sorted based on the respective metric. The x-axis represents percentile thresholds, the y-axis the percentage of low-quality sentences covered by the sentences below the respective threshold. The dashed line represents a theoretical baseline if low-quality sentence were distributed uniformly after shuffling the test set.

6.3 Low Translation Quality Experiment

Since a focus of our system is finding erroneous translations, we also analyzed the use of metrics for finding these translations in particular. Again, we used the test set from the Khresmoi data set, translating from German to English. In the context of this experiment, we consider translations with CharacTer score larger than one standard deviation above the median score of the test set to have low translation quality. The median score was 0.43, with a standard deviation of 0.18, thus we considered all translations with score > 0.61 to have low translation quality, which made up 179 sentences (17.9%) of the data set. First, the test set was sorted based on each metric, using descending order for all penalty metrics (CP, CDP, AP_{in}, AP_{out}) and length, and ascending order for confidence. We then iterated over each sentence of the sorted test set, recording the percentage of low-quality sentences covered so far after each test set sentence.

Figure 6.3 shows the percentage of low-quality translations covered depending on the percentage of all sentences under a percentile threshold for each evaluated metric. The results for confidence, CDP, AP_{in} and AP_{out} show similar behaviour. For these metrics, the cover percentage increases sharply at first, with 25% of low-quality translations covered after 7.5% (confidence), 8.9% (CDP),

6.7% (AP_{in}) and 10% (AP_{out}) of all sentences. In other words, a quarter of low-quality translations is found after considering 6-10% of all sentences sorted by these metrics. 50% of low-quality translations are covered after 21.2% (confidence), 24.2% (CDP), 20% (AP_{in}) and 23.7% (AP_{out}) of all sentences are considered. For 75% cover, just over 50% of all sentences must be considered for all four metrics, showing diminishing returns. The rate of increase then decreases towards 100% cover, which is achieved only after considering all sentences. Notably, CP and length perform similarly to the random baseline, meaning that sorting based on these metrics brings little benefit for finding low-quality translations sooner than shuffling sentences randomly. For example, 19.6% of sentences must be considered to cover 25% of low-quality sentences for CP, which is an improvement of only 5.4% compared to the random baseline.

6.4 Domain Adaption Experiment

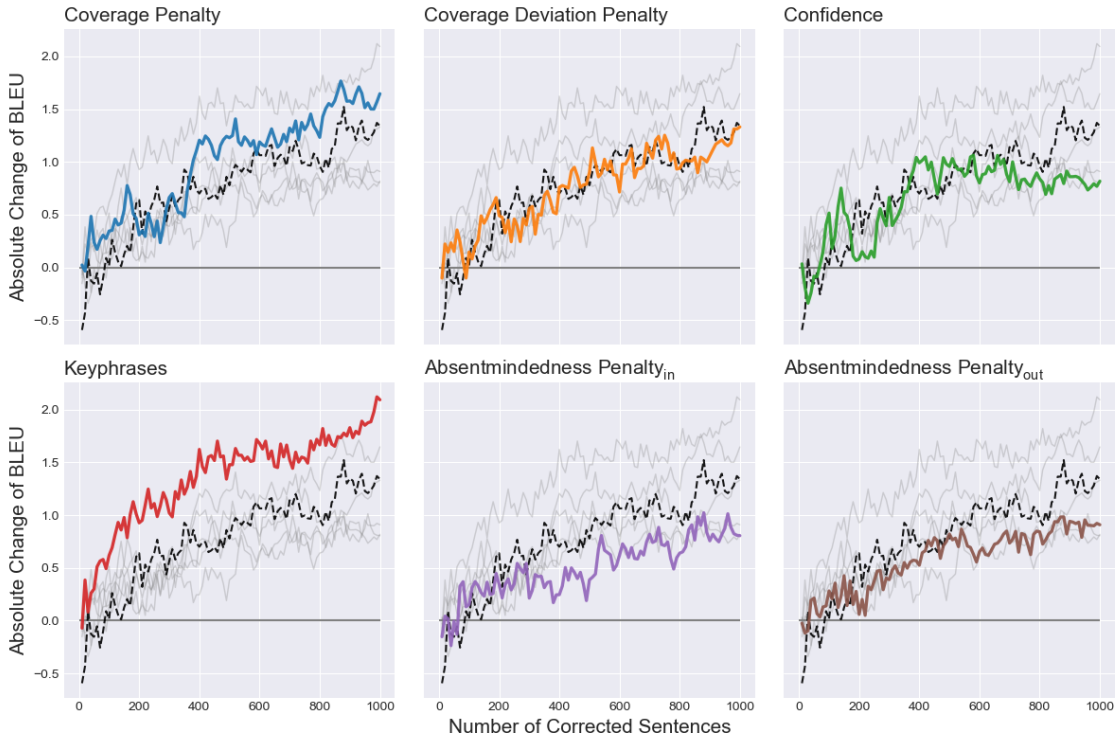

Figure 6.4: Absolute change of BLEU score during online domain adaption for each metric on Khresmoi medical test data compared to an offline system trained only on out-of-domain data. The x-axis shows the number of sentences corrected so far, the y-axis is the difference between the BLEU of the online system and the offline system on the hold-out set. For each metric, all sentences were sorted in descending order, except for confidence, where ascending order was used. The dashed line represents the baseline when sentences are shuffled randomly rather than sorted based on a metric.

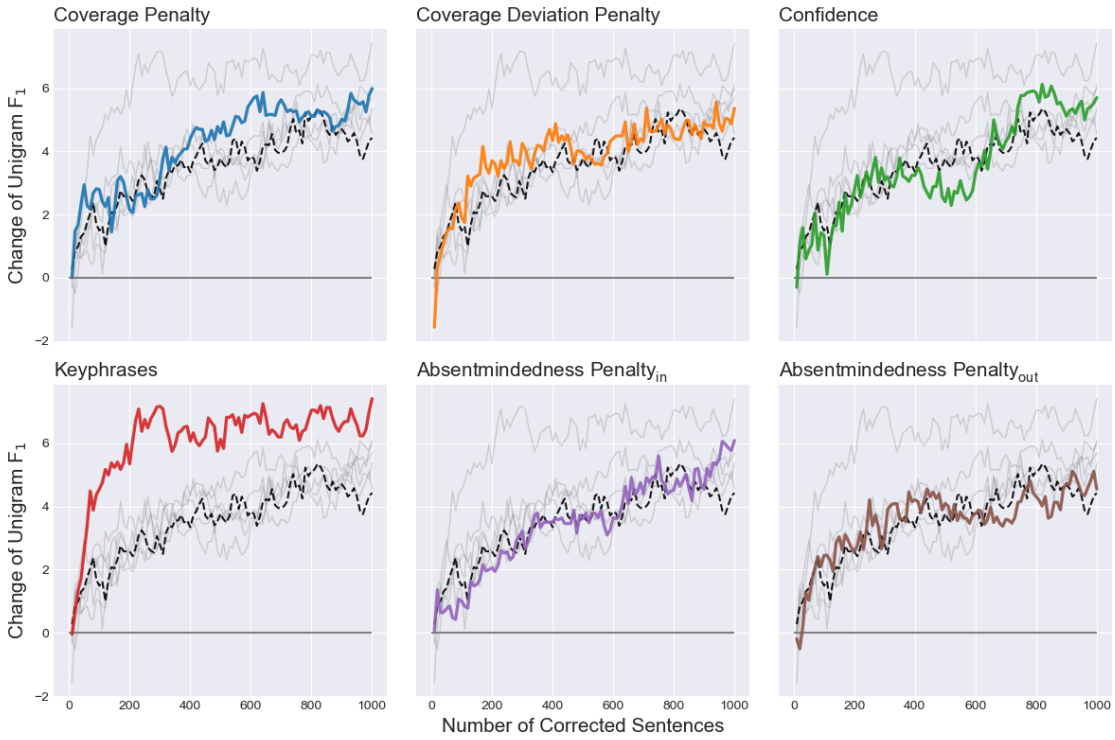

Figure 6.5: Absolute change of clipped unigram F_1 score of rare words during online domain adaption for each metric on Khresmoi medical test data compared to an offline system trained only on out-of-domain data. The x-axis shows the number of sentences corrected so far, the y-axis is the absolute change between the unigram precision of the offline system and the online system on the test sentences. For each metric, all sentences were sorted in descending order, except for confidence, where ascending order was used. The dashed line represents the baseline when sentences are shuffled randomly rather than sorted based on a metric.

6.4.1 Setup

Another important feature of the system is domain adaption: the ability to feed user corrections back into the NMT model, thus improving translation quality for domain-specific documents. We evaluated the proposed system by analyzing how different metrics affect the system's ability to incorporate corrections. Because conducting such an experiment in real post-editing scenario would be too costly, we simulate post-editing by replacing machine translations by reference translations, as was done by Peris et al. [PCC17]. We use the test set for online training and the development set as a holdout set to evaluate domain adaption. Note that we evaluated the keyphrase metric instead of the length metric as in the correlation analysis, as we want to analyze how correcting sentences with many keyphrases influences domain adaption.

We conducted the experiment separately for each metric. First, all sentences from the training set were sorted based on the current metric. For the confidence metric, ascending sort order was used to start training with the translations with the lowest confidence score. For all other metrics, descending sort order was used. Then, an online model pre-trained on out-of-domain data as

described in Section 4.10, was trained in an online fashion for each sentence in the specified order. After each sentence, the current online model was evaluated on the hold-out set. Additionally, the performance of an offline model on the holdout set was computed as a baseline. For online training, a fine-tuning approach as described in Section 4.8 was used. For evaluation on the holdout set, we computed BLEU scores and unigram F_1 of rare words. We also computed the performance for a random baseline, where sentences were first shuffled randomly, to find out not only whether certain metrics perform better than others for domain adaption but also whether using metrics is beneficial at all.

6.4.2 Translation Quality

First, we recorded how the translation quality as measured by BLEU score on the holdout set progressed during the domain adaption experiment for each metric. Figure 6.4 shows the absolute change in BLEU score for the medical document for the domain adaption experiment compared to the offline model. First, it appears that regardless of metric, the model successfully adapts to the domain with an increasing number of corrections, as seen by the steady increase in absolute improvement over the offline model. In agreement with Peris et al. [PCC17], we also find that the percentual change first behaves chaotically and declines or even becomes negative during the first sentences, likely due to the use of adaptive gradient optimization using Adam, which accumulates previous gradients for updating model weights. Confidence shows the largest initial decrease with -0.34 points after training on the first 20 sentences. Starting from about 50 sentences, however, the percentual change starts to improve steadily. The keyphrase metric performs the best, improving $+2.12$ BLEU points over the offline model after training on the first 980 sentences. Additionally, it is the only metric that clearly outperforms the random baseline over the complete run, with roughly $+1$ relative improvement over large parts of the experiment. CP achieves the second highest improvement, with $+1.77$ change in BLEU score after 860 sentences, however only slightly outperforming the random baseline which has a maximum improvement of $+1.52$. The remaining metrics confidence, AP_{in} and AP_{out} show maximum BLEU improvements of only $+1.06$, $+1.02$ and $+0.98$ respectively. Additionally, these metrics seem to perform equally or worse than the random baseline, with AP_{in} and AP_{out} showing lower improvement over the whole run and CDP with similar improvements. Notably, the improvement over the offline model starts decreasing for the confidence metrics after 500 sentences, finishing at less than $+1$ improvement of BLEU score.

6.4.3 Translation of Rare Words

We also analyzed whether domain adaption improved the translation of rare words. We computed the clipped unigram F_1 score of rare words, as described by Sennrich et al. [SHB15] as a metric for measuring rare word translation. For this experiment, all words in the document that did not appear in the WMT'16 training set were considered as rare. Again, we train a model in an online fashion for all 1000 sentence of the training set and compute the unigram F_1 score on the holdout set after each sentence. Then, the absolute difference between the unigram F_1 score of the online model and the offline model is recorded after each step.

Figure 6.5 shows the absolute differences for each metric, as well as a baseline where sentences are shuffled randomly before training. Regardless of metric, the model cleary improves regarding the translation of rare words over the offline model, as the unigram F_1 score increases after training on

Source	Multiple epiphysäre Dysplasie kann auch durch ein autosomal-rezessives Muster vererbt werden, was bedeutet , dass beide Kopien des Gens in jeder Zelle Mutationen aufweisen .
Reference	Multiple epiphyseal dysplasia can also be inherited in an autosomal recessive pattern, which means both copies of the gene in each cell have mutations.
Offline Model	Multiple epiphysical Dysplasie can also be inherited by an autosomal recessionary pattern, which means that both copies of the Gens have mutations in every cell.
Online Model	Multiple epiphysical dysplasia can also be inherited by an autosomal recessionary pattern, which means that both copies of the gene have mutations in each cell.

Table 6.2: The source sentence, reference translation, translation of the offline model, and the online model after training during the domain adaption experiment for a sample sentence from the Khresmoi medical data set. Rare words whose translations were improved in the online model are underlined. Rare words whose translations were not improved are underlined dashed.

an increasing number of sentences. Unlike the percentual change of BLEU scores, F_1 scores seem to not suffer as strongly from initial chaotic behaviour due to Adam. All metrics achieve at least +5 increase in unigram F_1 score, with the smallest improvement of +5.1 for AP_{out} and the largest improvement of up to +7.4 for the keyphrase metric. Notably, the results for the keyphrase metric differ the most from the other metrics. The unigram F_1 improves dramatically during the first 200 sentences, quickly reaching +6 improvement. The difference of improvement between keyphrase metric and the others, including the random baseline, is large with up to +4 improvement relative to other metrics. However, after about 200 sentences the improvement compared to the offline model fluctuates between +6 and +7.4, suggesting that a limit is reached. In the second place, CP performs slightly better throughout the experiment compared to the random baseline, bringing the second largest improvement of +7. The other metrics improve over the offline model steadily during training but do not seem to perform consistently better compared to the random baseline. AP_{in} and AP_{out} perform similarly to the random baseline, showing a slow but steady increase during the experiment. CDP and confidence show similar behaviour, with CDP starting with the largest decrease in unigram F_1 score but recovering quickly. Since the NMT model was trained on the same sentences for each metrics, with the only difference being sentence order, these results suggest that the order in which the NMT model is trained has an effect on online domain adaption of rare words.

In order to evaluate the improvement qualitatively sample sentences were analyzed where the online model improved rare word translation at the end of the experiment. Table 6.2 shows a reference translation, together with the translation of the offline model, trained only on out-of-domain data, and the online model trained during the domain adaption experiment. Note how the offline model translates the source words “Dysplasie” and “Gens” by literally copying them into the translation. On the other hand, the online model successfully learns to translate both domain-specific words into “dysplasia” and “gene”, respectively. However, neither model correctly translates the source

words “epiphysäre” and “rezessives”. The failure of the online model to translate these words may be explained due to the difference in frequency in the test set of these words. “Dysplasie” (7 occurrences) and “Gens” (5 occurrences) appear more often than “epiphysären” (2 occurrences) and “rezessives” (1 occurrence). This result suggests that domain adaption is impeded by rare words with few occurrences, i.e. words that are not only rare in regards to out-of-domain data but also rare in regards to in-domain data.

6.5 Discussion

This section will discuss the results of the automated evaluation and relate them to the proposed system.

6.5.1 Correlation of Metrics and Translation Quality

The correlation analysis showed that the relevant metrics can be used as a proxy for translation quality. A correlation of metric score and CharacTer score with a magnitude of Pearson coefficient of 0.46 for the confidence metric was found on the medical corpus. Correlations of similar strength were found for CDP, AP_{in} and AP_{out}. As the confidence metric is an aggregation of both *AP* metrics and CDP, it is unsurprising that its strength of correlation is comparable to the aforementioned metrics. Sentence length and CP showed only weak correlations with $r = 0.11$ and $r = 0.16$, respectively. Plotting the results revealed a large variance around the median of metric scores, implying that the metrics are not a good predictor of translation quality for these ranges. However, for high scores of penalty metrics and low scores of confidence variance is reduced. Therefore, extremely high or low scores for penalties and confidence respectively can be used as an indicator of low translation quality, whereas values around the median are less of an indicator of translation quality. The metric distributions also revealed numerous outliers for high scores in penalty metrics and low scores for confidence, again suggesting that erroneous translations are found for extreme metric scores.

Relating these results back to the proposed system, this suggests that the Metrics View can be useful for detecting outliers and to find sentences that are likely to contain errors, especially using the confidence metric, which has one of the strongest correlations with translation quality. The evaluation showed a weak correlation between length and translation quality but filtering sentences based on length might still be useful, e.g. for finding the longest or shortest sentences in a document. However, all sentences with translation errors cannot be found only by looking at outliers for a single metric. Because the Metrics View allows filtering based on multiple metrics at the same time, this problem might be mitigated in practice. Further experiments evaluating correlations of multiple metrics may be useful to gain more insight.

In a separate experiment, we showed that sorting sentences by confidence, CDP, AP_{in} and AP_{out} bring large gains for detecting low-quality translations, compared to correcting sentences in a random order. A quarter of low-quality translations can be found by considering less than 10% of sentences, and over half of low-quality translations can be found by considering around 25% of all sentences. On the other hand, sorting based on length and CP brings no measurable improvement over the random baseline. These results validate one of the main goals of NMTVis for reducing

post-editing and translation effort, as a majority of erroneous translations can thus be found faster by using the proposed metrics. However, in a scenario where all low-quality translations need to be found and corrected, all sentences need to be considered even when sorting based on metrics, meaning that the metrics alone cannot detect all translation errors.

6.5.2 Domain Adaption

The evaluation of domain adaption showed that in a simulated post-editing experiment, our NMT model successfully learns from corrections in an online fashion for a medical corpus. In general, the NMT model adapts to the given domain as seen through improved translation quality on a holdout set. Sorting based on the number of keyphrases and coverage penalty show the largest improvement in translation quality for our NMT model with increases of up +2 and +1.6 BLEU points, respectively. However, other metrics also bring moderate improvement of up to 1.5 BLEU points. Results also show that a minimum amount of sentences needs to be corrected for domain adaption to bring improvement, due to the use of adaptive optimization algorithms used which depend on previous gradients. For our system, this means that users must first correct a certain amount of sentences but can benefit from the domain adaption approach afterward.

The evaluation further showed that the translation of rare words is improved through online learning for all metrics. Sorting sentences based on the number of keyphrases brings the largest improvement for translation of rare words with an increase of unigram F_1 score of up to +7.4 compared to an offline model, but other metrics perform only slightly worse, with improvements of +4 to +6. Additionally, keyphrases require the fewest number of corrected sentences to bring the largest gains, validating the use of the keyphrase metric in the Metrics View, as well as the Keyphrase View for sentence selection. These results suggest that the NMT model adapts to a domain better and faster when first trained on many sentences containing frequent domain-specific words. However, a qualitative analysis of rare word translations showed that domain adaption is impeded by keyphrases that occur only a few times in a document. These rare words are not translated correctly even after online domain adaption.

The results from the domain adaption experiments validate the functionality of NMTVIS to incorporate user corrections of machine-generated translations into the NMT model, as it improves translation quality and rare word translation for domain-specific documents. This may reduce post-editing and translation effort as domain adaption can improve translations for sentences that were not corrected by the user. However, further experiments are needed to analyze performance on documents across different domains and languages to generalize the findings of our experiments.

7 Conclusion

This thesis introduced NMTVis, a novel VA approach for NMT that supports users in translation tasks for large, domain-specific documents through interactive visualizations. First, goals and tasks were identified to guide the development of the system. Different views and visualizations were developed to support the tasks and goals. A sentence selection mechanism using keyphrase search and parallel coordinates plot was developed for finding relevant sentences that are likely to require manual post-editing based on multiple metrics. For post-editing, dynamic graph visualizations and interaction techniques were developed to allow exploration of translation hypotheses generated during beam search decoding and attention weights. The system supports domain adaption through an online training approach using manual corrections of machine-generated translations. Additionally, an attentional encoder-decoder NMT model using subword units for German→English translation was trained and used as the base model of NMTVis. The NMT model was evaluated and compared to state-of-the-art systems, showing satisfactory translation quality as well as a potential for further improvements. The system was implemented prototypically as a web application.

In order to validate the proposed system, a user study and an automated evaluation were conducted. The web-based user study, involving 15 researchers and students with visualization and natural language processing backgrounds, showed positive feedback in regards to effectiveness, ease of interaction and intuitiveness of visualizations of NMTVis. Participants showed a clear preference to use NMTVis over traditional text-based post-editing systems, especially for large documents. As a result of the qualitative feedback, multiple challenges were identified including scalability of visualizations and efficiency of interactions. Participants also noted potential improvements regarding interaction and novel features that could be integrated into NMTVis in the future to make it into a productivity tool for post-editing. The statistical evaluation showed that using the proposed metrics, particularly the confidence metric, can be effective for sentence selection due to correlations of metric scores and translation quality. A majority of low-quality translations can be found faster by sorting sentences based on the metrics compared to a random baseline, reducing post-editing effort. Additionally, a domain adaption experiment on a medical corpus showed that the domain adaption mechanism of NMTVis improves translation quality for (1) the whole document and (2) for rare words, which validates the use of the proposed system for domain-specific documents. The proposed keyphrase metric was found to bring the largest improvement for domain adaption over a random baseline.

In a broader context, this thesis showed that a VA approach for a specific Deep Learning task, such as machine translation, is a viable way to make complex, opaque models and their predictions more understandable for human users, and that VA can be a useful tool to bridge the gap between users and Deep Learning models. However, more research is needed to study how visualizations can make models more explainable and interpretable. Additionally, benchmarks and data sets are necessary to evaluate the impact of VA systems for Deep Learning quantitatively and to enable comparison between different systems. Explainability in Deep Learning through visualizations remains a highly relevant future research topic with many opportunities and challenges.

Outlook

For future work, a wide range of research directions could be pursued:

User Study A quantitative user study should be conducted, focusing on the effectiveness of the system for post-editing regarding (1) translation quality and (2) translation speed. This could be done by comparing the system against a baseline system where only text-editing is possible. Previous post-editing studies such as [GHM13] should be considered for study design and evaluation metrics. Lagarda et al. [LAC+09] evaluated a statistical machine translation system by recording users during translation and post-editing, measuring several metrics such as edit time, number of edits, translation throughput, productivity, gain and edit distance, which could be used as a basis for a future user study.

NMT Model Due to the rapid progress in NMT research, novel approaches and architectures could be incorporated into the system to further improve performance. One such example is the recent Transformer model architecture, introduced by Vaswani et al. [VSP+17] which uses multi-headed self-attention instead of an encoder-decoder RNN architecture. This novel attention mechanism might also lead to better insights about the model and could be visualized to the user.

Languages This thesis focused on German→English translation for the prototypical implementation and evaluation of the system. Both languages use the latin alphabet and share many linguistic similarities. However, logographic writing systems, for example, may pose a challenge for interactive post-editing using beam search. Therefore, other languages should be incorporated into the system in order to validate the approach regarding translation from or to languages with different writing systems such as Chinese.

Domain Adaption The proposed system used a simple fine-tuning approach for domain adaption using manual corrections, where model weights are frozen and corrections are integrated in an online fashion. More sophisticated approaches for domain adaption exist [PNW18], that might improve translation quality for specific domains. Instance weighting [WUL+17] in particular may be a promising technique for adapting the model to users' corrections, allowing the model to improve on domain-specific documents while also maintaining translation performance on out-of-domain data.

Use Cases This thesis focused on machine translation as the main use case for the proposed system. Different sequence-to-sequence tasks, such as text summarization, question answering or image captioning could be integrated into the system to support users in a wide range of tasks. New interaction and visualization approaches would be needed to handle different data types such as images, audio, and video.

Scalability of Visualizations The user study showed that large documents and long sentences proved to be challenging when using the interactive visualizations implemented in the system. More thought must be put into how these visualizations can be scaled up, as long sentences and large documents are use cases where the proposed system could help users the most for post-editing. For the Beam Search View, clustering of nodes that can be collapsed or expanded through interaction might be a useful extension to handle long translation hypotheses.

Bibliography

[APS16] M. Allamanis, H. Peng, C. Sutton. "A Convolutional Attention Network for Extreme Summarization of Source Code". In: *International Conference on Machine Learning*. 2016, pp. 2091–2100 (cit. on p. 32).

[BCB14] D. Bahdanau, K. Cho, Y. Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: *arXiv preprint arXiv:1409.0473* (2014) (cit. on pp. 28, 30, 54).

[BCF+16] O. Bojar, R. Chatterjee, C. Federmann, Y. Graham, B. Haddow, M. Huck, A. J. Yepes, P. Koehn, V. Logacheva, C. Monz, et al. "Findings of the 2016 Conference on Machine Translation." In: *ACL 2016 FIRST CONFERENCE ON MACHINE TRANSLATION (WMT16)*. The Association for Computational Linguistics. 2016, pp. 131–198 (cit. on p. 58).

[CGCB14] J. Chung, Ç. Gülçehre, K. Cho, Y. Bengio. "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling". In: *CoRR* abs/1412.3555 (2014). arXiv: 1412.3555. URL: http://arxiv.org/abs/1412.3555 (cit. on p. 18).

[CHC+16] S. Cheng, S. Huang, H. Chen, X.-Y. Dai, J. Chen. "PRIMT: A Pick-Revise Framework for Interactive Machine Translation". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, 2016, pp. 1240–1249. DOI: 10.18653/v1/N16-1148. URL: http://www.aclweb.org/anthology/N16-1148 (cit. on p. 24).

[CMG+14] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, Y. Bengio. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: *CoRR* abs/1406.1078 (2014). arXiv: 1406.1078. URL: http://arxiv.org/abs/1406.1078 (cit. on p. 27).

[CNS+14] M. Cettolo, J. Niehues, S. Stüker, L. Bentivogli, M. Federico. "Report on the 11th IWSLT evaluation campaign, IWSLT 2014". In: *Proceedings of the International Workshop on Spoken Language Translation, Hanoi, Vietnam*. 2014 (cit. on p. 57).

[CT05] K. A. Cook, J. J. Thomas. *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. Tech. rep. Pacific Northwest National Lab.(PNNL), Richland, WA (United States), 2005 (cit. on p. 19).

[CVG+14] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio. "Learning phrase representations using RNN encoder-decoder for statistical machine translation". In: *arXiv preprint arXiv:1406.1078* (2014) (cit. on p. 18).

[CW18] C. Chu, R. Wang. "A Survey of Domain Adaptation for Neural Machine Translation". In: *arXiv preprint arXiv:1806.00258* (2018) (cit. on pp. 11, 34).

[Dav89] F. D. Davis. “Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology”. In: *MIS Quarterly* 13.3 (1989), pp. 319–340. ISSN: 02767783. URL: http://www.jstor.org/stable/249008 (cit. on p. 64).

[DHH+17] O. Dušek, J. Hajič, J. Hlaváčová, J. Libovický, P. Pecina, A. Tamchyna, Z. Urešová. *Khresmoi Summary Translation Test Data 2.0*. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University. 2017. URL: http://hdl.handle.net/11234/1-2122 (cit. on p. 71).

[DN17] M. J. Denkowski, G. Neubig. “Stronger Baselines for Trustable Results in Neural Machine Translation”. In: *CoRR* abs/1706.09733 (2017). arXiv: 1706.09733. URL: http://arxiv.org/abs/1706.09733 (cit. on pp. 58, 61).

[FWP+14] M. Freitag, J. Wuebker, S. Peitz, H. Ney, M. Huck, A. Birch, N. Durrani, P. Koehn, M. Mediani, I. Slawik, et al. “Combined spoken language translation”. In: *Proc. of the Int. Workshop on Spoken Language Translation (IWSLT), South Lake Tahoe, CA, USA*. 2014 (cit. on p. 61).

[GBC16] I. Goodfellow, Y. Bengio, A. Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016. ISBN: 9780262035613 (cit. on p. 13).

[GHM13] S. Green, J. Heer, C. D. Manning. “The Efficacy of Human Post-editing for Language Translation”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’13. Paris, France: ACM, 2013, pp. 439–448. ISBN: 978-1-4503-1899-0. DOI: 10.1145/2470654.2470718. URL: http://doi.acm.org/10.1145/2470654.2470718 (cit. on p. 82).

[Gra12] A. Graves. “Sequence Transduction with Recurrent Neural Networks”. In: *CoRR* abs/1211.3711 (2012). arXiv: 1211.3711. URL: http://arxiv.org/abs/1211.3711 (cit. on p. 31).

[HKPC18] F. Hohman, M. Kahng, R. Pienta, D. H. Chau. “Visual Analytics in Deep Learning: An Interrogative Survey for the Next Frontiers”. In: *arXiv preprint arXiv:1801.06889* (2018) (cit. on pp. 20, 21, 37).

[HL17] C. Hokamp, Q. Liu. “Lexically Constrained Decoding for Sequence Generation Using Grid Beam Search”. In: *CoRR* abs/1704.07138 (2017). arXiv: 1704.07138. URL: http://arxiv.org/abs/1704.07138 (cit. on pp. 24, 30).

[HS97] S. Hochreiter, J. Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9.8 (1997), pp. 1735–1780 (cit. on p. 17).

[HSW89] K. Hornik, M. Stinchcombe, H. White. “Multilayer feedforward networks are universal approximators”. In: *Neural Networks* 2.5 (1989), pp. 359–366 (cit. on p. 13).

[Hut05] J. Hutchins. “Current commercial machine translation systems and computer-based translation tools: system types and their uses”. In: *International Journal of Translation* 17.1-2 (2005), pp. 5–38 (cit. on p. 11).

[Ins85] A. Inselberg. “The plane with parallel coordinates”. In: *The Visual Computer* 1.2 (1985), pp. 69–91. DOI: 10.1007/BF01898350. URL: https://doi.org/10.1007/BF01898350 (cit. on p. 42).

[KAF+08] D. Keim, G. Andrienko, J.-D. Fekete, C. Görg, J. Kohlhammer, G. Melançon. "Visual Analytics: Definition, Process, and Challenges". In: *Information Visualization: Human-Centered Issues and Perspectives*. Ed. by A. Kerren, J. T. Stasko, J.-D. Fekete, C. North. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 154–175. ISBN: 978-3-540-70956-5. DOI: 10.1007/978-3-540-70956-5_7. URL: https://doi.org/10.1007/978-3-540-70956-5_7 (cit. on p. 19).

[KB14] D. P. Kingma, J. Ba. "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980 (2014). arXiv: 1412.6980. URL: http://arxiv.org/abs/1412.6980 (cit. on pp. 24, 58).

[KK16] R. Knowles, P. Koehn. "Neural Interactive Translation Prediction". In: *Proceedings of the Association for Machine Translation in the Americas*. 2016, pp. 107–120 (cit. on p. 24).

[KK17] P. Koehn, R. Knowles. "Six Challenges for Neural Machine Translation". In: *CoRR* abs/1706.03872 (2017). arXiv: 1706.03872. URL: http://arxiv.org/abs/1706.03872 (cit. on pp. 11, 54, 58).

[KKE10] E. D. Keim, J. Kohlhammer, G. Ellis. "Mastering the Information Age: Solving Problems with Visual Analytics, Eurographics Association". In: (2010) (cit. on pp. 19, 63).

[LAC+09] A.-L. Lagarda, V. Alabau, F. Casacuberta, R. Silva, E. Dıaz-de-Liaño. "Statistical Post-editing of a Rule-based Machine Translation System". In: *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*. NAACL-Short '09. Boulder, Colorado: Association for Computational Linguistics, 2009, pp. 217–220. URL: http://dl.acm.org/citation.cfm?id=1620853.1620913 (cit. on p. 82).

[Lew95] J. R. Lewis. "IBM computer usability satisfaction questionnaires: Psychometric evaluation and instructions for use". In: *International Journal of Human–Computer Interaction* 7.1 (1995), pp. 57–78. DOI: 10.1080/10447319509526110. URL: https://doi.org/10.1080/10447319509526110 (cit. on p. 64).

[Lip15] Z. C. Lipton. "A Critical Review of Recurrent Neural Networks for Sequence Learning". In: *CoRR* abs/1506.00019 (2015). arXiv: 1506.00019. URL: http://arxiv.org/abs/1506.00019 (cit. on p. 16).

[LPM15] M. Luong, H. Pham, C. D. Manning. "Effective Approaches to Attention-based Neural Machine Translation". In: *CoRR* abs/1508.04025 (2015). arXiv: 1508.04025. URL: http://arxiv.org/abs/1508.04025 (cit. on pp. 27, 29, 56, 57, 61).

[LSK17] J. Lee, J.-H. Shin, J.-S. Kim. "Interactive Visualization and Manipulation of Attention-based Neural Machine Translation". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. 2017, pp. 121–126 (cit. on pp. 11, 25, 47).

[Lun01] A. M. Lund. "Measuring Usability with the USE Questionnaire". In: *Usability Interface* 8.2 (2001), pp. 3–6 (cit. on p. 64).

[MRO+12] A. M. MacEachren, R. E. Roth, J. O'Brien, B. Li, D. Swingley, M. Gahegan. "Visual Semiotics & Uncertainty Visualization: An Empirical Study". In: *IEEE Transactions on Visualization and Computer Graphics* 18.12 (Dec. 2012), pp. 2496–2505. ISSN: 1077-2626. DOI: 10.1109/TVCG.2012.279 (cit. on p. 49).

[Ola15] C. Olah. *Understanding LSTM Networks*. http://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Online; accessed 02-August-2018]. 2015 (cit. on p. 17).

[PCC17] Á. Peris, L. Cebrián, F. Casacuberta. "Online Learning for Neural Machine Translation Post-editing". In: *CoRR* abs/1706.03196 (2017). arXiv: 1706.03196. URL: http://arxiv.org/abs/1706.03196 (cit. on pp. 24, 34, 76, 77).

[PNW18] N.-Q. Pham, J. Niehues, A. Waibel. "Towards one-shot learning for rare-word translation with external experts". In: *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*. 2018, pp. 100–109 (cit. on p. 82).

[PRWZ02] K. Papineni, S. Roukos, T. Ward, W.-J. Zhu. "BLEU: A Method for Automatic Evaluation of Machine Translation". In: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. ACL '02. Philadelphia, Pennsylvania: Association for Computational Linguistics, 2002, pp. 311–318. DOI: 10.3115/1073083.1073135. URL: https://doi.org/10.3115/1073083.1073135 (cit. on p. 35).

[Rei00] U.-D. Reips. "Chapter 4 - The Web Experiment Method: Advantages, Disadvantages, and Solutions". In: *Psychological Experiments on the Internet*. Ed. by M. H. Birnbaum. San Diego: Academic Press, 2000, pp. 89–117. ISBN: 978-0-12-099980-4. DOI: https://doi.org/10.1016/B978-012099980-4/50005-8. URL: http://www.sciencedirect.com/science/article/pii/B9780120999804500058 (cit. on p. 63).

[RFB17] M. Rikters, M. Fishel, O. Bojar. "Visualizing Neural Machine Translation Attention and Confidence". In: *The Prague Bulletin of Mathematical Linguistics* 109.1 (2017), pp. 39–50. URL: https://content.sciendo.com/view/journals/pralin/109/1/article-p39.xml (cit. on pp. 11, 25, 43).

[RHW86] D. E. Rumelhart, G. E. Hinton, R. J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323.6088 (1986), p. 533. URL: http://dx.doi.org/10.1038/323533a0 (cit. on p. 15).

[RT81] E. M. Reingold, J. S. Tilford. "Tidier Drawings of Trees". In: *IEEE Transactions on Software Engineering* 2 (1981), pp. 223–228 (cit. on p. 56).

[Sch97] R. J. Schalkoff. *Artificial Neural Networks*. Vol. 1. McGraw-Hill New York, 1997. ISBN: 978-0070571181 (cit. on p. 13).

[SDS+06] M. Snover, B. Dorr, R. Schwartz, L. Micciulla, J. Makhoul. "A Study of Translation Edit Rate with Targeted Human Annotation". In: *Proceedings of Association for Machine Translation in the Americas*. Vol. 200. 6. 2006 (cit. on p. 35).

[SGB+18] H. Strobelt, S. Gehrmann, M. Behrisch, A. Perer, H. Pfister, A. M. Rush. "Seq2Seq-Vis: A Visual Debugging Tool for Sequence-to-Sequence Models". In: *CoRR* abs/1804.09299 (2018). arXiv: 1804.09299. URL: http://arxiv.org/abs/1804.09299 (cit. on pp. 11, 25, 47, 54).

[SGPR18] H. Strobelt, S. Gehrmann, H. Pfister, A. M. Rush. "LSTMVis: A Tool for Visual Analysis of Hidden State Dynamics in Recurrent Neural Networks". In: *IEEE Transactions on Visualization and Computer Graphics* 24.1 (Jan. 2018), pp. 667–676. ISSN: 1077-2626. DOI: 10.1109/TVCG.2017.2744158 (cit. on pp. 11, 24).

[SHB15] R. Sennrich, B. Haddow, A. Birch. "Neural Machine Translation of Rare Words with Subword Units". In: *CoRR* abs/1508.07909 (2015). arXiv: 1508.07909. URL: http://arxiv.org/abs/1508.07909 (cit. on pp. 32, 33, 57, 60, 77).

[SHB16] R. Sennrich, B. Haddow, A. Birch. "Edinburgh Neural Machine Translation Systems for WMT 16". In: *CoRR* abs/1606.02891 (2016). arXiv: 1606.02891. URL: http://arxiv.org/abs/1606.02891 (cit. on pp. 11, 58, 61).

[SLM17] A. See, P. J. Liu, C. D. Manning. "Get To The Point: Summarization with Pointer-Generator Networks". In: *CoRR* abs/1704.04368 (2017). arXiv: 1704.04368. URL: http://arxiv.org/abs/1704.04368 (cit. on p. 28).

[SVL14] I. Sutskever, O. Vinyals, Q. V. Le. "Sequence to Sequence Learning with Neural Networks". In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, K. Q. Weinberger. Curran Associates, Inc., 2014, pp. 3104–3112. URL: http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf (cit. on p. 32).

[TLS+17] Z. Tu, Y. Liu, L. Shang, X. Liu, H. Li. "Neural Machine Translation with Reconstruction". In: *AAAI*. 2017, pp. 3097–3103 (cit. on p. 11).

[VSP+17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin. "Attention is All you Need". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett. Curran Associates, Inc., 2017, pp. 5998–6008. URL: http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf (cit. on pp. 11, 82).

[WPRN16] W. Wang, J.-T. Peter, H. Rosendahl, H. Ney. "CharacTER: Translation Edit Rate on Character Level". In: *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*. Vol. 2. 2016, pp. 505–510 (cit. on p. 36).

[WSC+16] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, J. Dean. "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation". In: *CoRR* abs/1609.08144 (2016). arXiv: 1609.08144. URL: http://arxiv.org/abs/1609.08144 (cit. on pp. 11, 31, 43, 57).

[WUL+17] R. Wang, M. Utiyama, L. Liu, K. Chen, E. Sumita. "Instance Weighting for Neural Machine Translation Domain Adaptation". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 2017, pp. 1482–1488 (cit. on p. 82).

[YNN+17] A. J. Yepes, A. Névéol, M. Neves, K. Verspoor, O. Bojar, A. Boyer, C. Grozea, B. Haddow, M. Kittner, Y. Lichtblau, et al. "Findings of the WMT 2017 Biomedical Translation Shared Task". In: *Proceedings of the Second Conference on Machine Translation*. 2017, pp. 234–247 (cit. on p. 71).

All links were last followed on September 25, 2018.

A Questionnaire

General

What is your occupation?. . . . . | Researcher | Student | Other |

Are you affiliated with a research institute?. . . . . | IMS | VIS/VISUS | None | Other |

Background

	← None						Expert Knowledge →
How much knowledge do you have about Machine Translation? . . .	1	2	3	4	5	6	7
How much knowledge do you have about Visualizations?	1	2	3	4	5	6	7

Language Skills

	← Poor						Native Speaker →
How do you rate your German language proficiency?	1	2	3	4	5	6	7
How do you rate your English language proficiency?	1	2	3	4	5	6	7

Functionality

	← strongly disagree						agree completely →
Overall, I am satisfied with the ease of completing the tasks in this scenario.	1	2	3	4	5	6	7
I would prefer the system over a simple text field for translating a large document (>100 sentences).	1	2	3	4	5	6	7
I would prefer the system over a simple text field for translating a small document (<20 sentences).	1	2	3	4	5	6	7

Metrics View

	← strongly disagree						agree completely →
The Metrics View was useful for finding sentences that contain translation errors.	1	2	3	4	5	6	7
It was easy to understand the visual representations in the Metrics View.	1	2	3	4	5	6	7
It was easy to interact with the Metrics View.	1	2	3	4	5	6	7

Keyphrase View

	agree completely → ← strongly disagree						
The Keyphrase View was useful for finding sentences that contain translation errors.	1	2	3	4	5	6	7
It was easy to understand the visual representations in the Keyphrase View.	1	2	3	4	5	6	7
It was easy to interact with the Keyphrase View.	1	2	3	4	5	6	7

Beam Search View

	agree completely → ← strongly disagree						
The Beam Search View was helpful for exploring and correcting translations.	1	2	3	4	5	6	7
It was easy to understand the visual representations in the Beam Search View.	1	2	3	4	5	6	7
It was easy to interact with the Beam Search View.	1	2	3	4	5	6	7

Attention View

	agree completely → ← strongly disagree						
The Attention View was helpful for analysing translations.	1	2	3	4	5	6	7
It was easy to understand the visual representations in the Attention View.	1	2	3	4	5	6	7
It was easy to interact with the Attention View.	1	2	3	4	5	6	7

Feedback

What functionality or part of the system did you like and why?

What functionality or part of the system did you dislike and why?

Do you have suggestions for improvements to be made for the system?

Do you have any other feedback or comments?

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature