Can oracle-based imitation learning improve neural machine translation with data aggregation?

—

Kann Orakel-gestütztes Lernen durch Imitation mit Datenaggregation neuronale maschinelle Übersetzung verbessern?

Master's Thesis

Luca Oliver Hormann
3587152

At the Faculty of Mathematics and Computer Science
Institute of Computer Science

Reviewer:	Prof. Dr. Stefan Riezler
Supervisor:	Prof. Dr. Artem Sokolov

03.05.2021

Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Sowohl inhaltlich als auch wörtlich entnommene Inhalte wurden als solche kenntlich gemacht. Die Arbeit ist in gleicher oder vergleichbarer Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

Heidelberg, 03.05.2021 ____________________________

Luca Oliver Hormann

Acknowledgements

Throughout the writing of this thesis I have received great amounts of support and assistance.
I would first like to thank the state of Baden-Württemberg for the support of computational resources through the bwHPC cluster.
I thankfully acknowledge my supervisor, Dr. Artem Sokolov, whose expertise was invaluable when designing algorithms and objectives used to create the methods of this work.
I would like to thank Andrea Glaser, Michael Simon and Amadea Hofstötter for their proofreading and fruitful comments regarding the writing of this thesis.
Finally, I grateful to my family and friends because they have provided the social balance during this time as well as nice distractions to rest my mind outside of the research.

Abstract

Through globalization of the industry and the collaboration of nations world wide the importance of machine translated documents is rising. Many new ideas and concepts were proposed in the recent years to improve the overall quality of machine translation (MT). A lot of focus in machine learning is going towards the research of alternative learning techniques, as the basic existing paradigms such as supervised, unsupervised and reinforcement learning are not a perfect fit for every task. Imitation learning is a technique which combines the exploratory aspect of reinforcement with the efficiency of supervised learning. Through the usage of an interactive expert, the learning model, also called *student*, is able to obtain intermediate feedback for it's predictions at any given point in time. Thereby the student is aware of it's mistakes by considering the difference of it's and the expert's prediction from this point. Imitation learning can improve all major problems for training artificial neural networks given the right expert. These are: the time it takes to train a model, the acquisition of data and most importantly the overall performance given some specific metric.

The most common quality evaluation metric used for MT is BLEU. It is based on the n-gram precision between the generated translation, given some input, and the reference translation. Therefore it is non-differentiable and can not be used as a loss to train a MT model directly. For the training usually the maximum likelihood estimation (MLE) is used, such that the likelihood of each token in the output, given the input sequence, is maximized. This creates a discrepancy between training (MLE) and validation objective (BLEU). This thesis tries to overcome this issue by directly learning on the differences of the expected BLEU from the student and the expert in an imitation learning scenario. The expert is represented by a traditional statistical machine translation (SMT) model that should help the student in solving the problems mentioned above. For this a novel data aggregation method **A**ggregate **D**ata using approximated **BLEU** explorations (ADBLEU) based on imitation learning was implemented. After conducting several experiments, validating different approaches of data aggregation, it is shown that it is not possible to significantly improve the state-of-the-art student, due to limitations of the SMT expert.

Zusammenfassung

Durch die zunehmende industrielle Globalisierung und der daraus einhergehenden internationalen Zusammenarbeit steigt die Bedeutung von maschinell übersetzten Dokumenten. In den letzten Jahren wurden viele neue Ideen und Konzepte vorgeschlagen, um die Gesamtqualität der maschinellen Übersetzung zu verbessern. Ein Schwerpunkt im Bereich des maschinellen Lernens liegt in der Erforschung alternativer Lerntechniken. Das hat den Hintergrund, dass die bestehenden Paradigmen, wie überwachtes, unüberwachtes und verstärkendes Lernen nicht für jede Aufgabe optimal geeignet sind. Eine Technik, die den explorativen Aspekt des Verstärkungslernens mit der Effizienz des überwachten Lernens kombiniert, ist das Lernen durch Imitation. Durch den Einsatz eines interaktiven „Lehrer" ist der „Schüler" in der Lage, zu jedem Zeitpunkt, ein Zwischenfeedback für seine Vorhersagen zu erhalten. Durch die Abweichung seines Lösungsvorschlages, zu dem des Lehrers, wird dem Schüler sein eigener Fehler bewusst. Dadurch können essenzielle Probleme, wie zum Beispiel die Trainingszeit, die Datenerfassung und die Gesamtleistung anhand einer bestimmten Metrik, bei dem Training von künstlichen neuronalen Netzen, verbessert werden.
Die gebräuchlichste Metrik zur Qualitätsbewertung bei der maschinellen Übersetzung ist BLEU. Sie basiert auf der Präzision der N-Gramme zwischen der generierten Übersetzung und der Referenzübersetzung. Dadurch ist BLEU nicht differenzierbar und kann auch nicht direkt als Verlustfunktion für das Training in der maschinellen Übersetzung verwendet werden. In der Regel wird für das Training die Maximum-Likelihood-Schätzung verwendet, so dass die Wahrscheinlichkeit jedes Tokens in der Ausgabesequenz, entsprechend der Eingabesequenz, maximiert wird. Dadurch entsteht eine Diskrepanz zwischen Trainings- und Bewertungsziel. In dieser Arbeit wird dieses Problem versucht zu umgehen. In einem Imitationslernszenario wird der Schüler direkt durch Unterschiede des erwarteten BLEU-Ergebnisses zwischen sich und dem Lehrer trainiert. Der Lehrer wird durch ein SMT-Modell (Statistical Machine Translation) dargestellt, das dem Studenten bei den genannten Problemen helfen soll. Hierfür wurde eine neue Methode basierend auf dem Imitationslernen entwickelt: Das **A**ggregieren von **D**aten mit angenäherten **BLEU**-Explorationen (ADBLEU). Die durchgeführten Experimente, in denen verschiedene Ansätze der Datenaggregation validiert wurden, zeigen, dass es aufgrund von Einschränkungen bei dem SMT-Lehrers nicht möglich ist, den State-of-the-Art-Schüler signifikant zu verbessern.

Contents

List of Figures

List of Tables

1. Introduction

1.1. Motivation

The worldwide demand for communication between nations with different languages for the globalization of industry and the cultural exchange is constantly increasing. As a result, the requirements for Machine Translation (MT) have also been growing. Recent advances in Neural Machine Translation (NMT) with, for the subject, unconventional machine learning paradigms such as Reinforcement Learning (RL) have shown great success (Bahdanau, Brakel, et al., 2017; Ranzato et al., 2016; Wu et al., 2017), where systems that generate state-of-the-art performance are further improved using the exploratory nature of RL. Traditionally RL comes from the subject of control theory where either the environment or the rewards are unknown. The reward can usually be defined as a function that is based on actions and states. Doing an action in a certain state will yield a reward either immediately or at a later point in time. To understand what action in which state was good and which was bad, the accumulated reward at the end of the sequence of actions and states has to be assigned to individual steps. Therefore many different actions are taken to obtain knowledge of how the reward at the end of the sequence changes. In complex environments with millions of state-action pairs it can be tedious to learn how to obtain the highest reward from every state. Asking an expert that knows the environment and how to maximize the reward from every state sounds like a logical approach. The expert might be interactively queried to give an estimation of how to optimally continue the sequence given the current state. For instance in machine translation this expert might be a human that helps the system, also called *student*, to complete partially finished translations. Through this the student iteratively learns how to complete partially created translations in any state through data aggregation. This approach of learning is a part of Imitation Learning (IL), which can be seen as a machine learning paradigm between supervised and reinforcement learning. Because MT systems need large amounts of data using humans to complete their partial translations is infeasible. Therefore in this work a Statistical Machine Translation (SMT) system was used as an expert. SMT systems were mainly used before it was possible to train all of the free parameters that a NMT model needs to perform well. To output translations they learn the correspondences between source and target language by training on coocurrences of phrases in large amounts of aligned text. Typically SMT systems perform worse than NMT but are faster to train and to generate translations. To test if a state-of-the-art NMT system can be improved using IL with an interactive SMT expert the experiments described in this work were conducted.

1.2. Purpose and Research Question

Consequently, it is from major interest if and how a NMT model can be revised using imitation learning. Therefore, one of the most dominant questions is:

> **Can a state-of-the-art neural machine translation student be improved by using a statistical machine translation expert in an imitation learning scenario?**

This thesis will answer this research question. The main purpose is therefore to show if the hypothesis in question is possible and how it can be achieved. The trained NMT model will be tested in various metrics that are intended to represent different components of successful machine translation. This is done to show detailed insight of improvements in for example robustness when shifting the domain of the text, fluency of the generated text, generalization to unseen data and human judgement of the translation quality.

1.3. Approach

This work focuses on showing if it is possible to improve the performance of a state-of-the-art NMT system using an inferior sub-optimal SMT expert and if it is, what is needed to do it. To answer this question the implementation of an imitation learning framework for the integration of the SMT expert and a NMT student was done. The global objective for every MT system is to generate good translations on unseen data measured in BLEU, a common non-differentiable MT quality estimation metric.
First a dataset was selected that was flexible, fast to test on and still representable for the general case of MT. An existing SMT expert was modified for the usage in an IL context and adapted for the framework. The expert interfaces with the framework through word translation lattices that are weighted using an approximation of BLEU. Several experiments were done on the dataset to validate the overall performance of the expert and to show that the performance of it is usable for this scenario. The data aggregation imitation learning algorithm **A**ggregate **D**ata using approximated **BLEU** explorations (ADBLEU) was constructed using the fundamental ideas from previous works (Daumé et al., 2009; Ross & Bagnell, 2014; Ross, Munoz, et al., 2011). To test this method, a state-of-the-art baseline student was trained. The method aggregates data by using partial translations and exploratory actions of the student and continuing it with the expert. On a basic level this data is used to guide the student towards the best possible translations with regards to the expert. The student can infer a loss between it's and the expert's belief how to continue the sentence. Therefore the student is directly optimized for the global objective. Various ways of data aggregation were tested using different exploration methods. In the end the tests to answer the main research question were conducted using the best possible parameters for the method obtained from previous evaluations.

1.4. Scope and Limitations

Due to the fixed time schedule of this project some limitations concerning the dataset size and the amount of experiments had to be done. With the research question in mind a smaller, more flexible dataset was chosen, such that experiments could be done in quick succession. This helped to quickly find problems in the implementation and for the optimization of the hyperparameters. On the other hand by using this smaller dataset, not all capabilities of the method are fully represented. Applying a larger dataset on the same problem could yield similar but different results. As this work is mainly focused on showing if it is possible to improve a NMT student using a SMT expert, it was not necessary to further experiments or use larger datasets. Nevertheless in section 7.2 suggestions for improvements of the method and it's scope are done for future work.

1.5. Outline

This thesis will start with the theoretical foundations, evaluation metrics for MT and basic machine learning paradigms in chapter 2. The fundamentals shown in the theoretical part will be used in the following chapters and therefore should be understood by the reader. In chapter 3 the concept of the work is related to other methods that either try to solve a similar problem with a different method or use a similar method to solve a different problem. It is therefore put into a research perspective. Then the main methodology of the work follows in chapter 4. There the concept, algorithm and objectives are shown to the reader in detail. After that the results of the proposed method and the comparison to other MT systems in chapter 5 are presented. At last the results are analyzed in detail in chapter 6 and a conclusion is drawn to answer the research question and reveal options for future work in chapter 7.

2. Theoretical Foundations

In this chapter the basics of machine translation will be shown. This includes statistical and neural machine translation with both it's benefits and drawbacks. Furthermore, Moses, a statistical machine translation toolkit will be reviewed. As a common way of evaluating translations the n-gram based score BLEU is introduced. Additionally different paradigms of machine learning are considered in the scope of decisions making. Finally decision processes will be reviewed in more detail and illustrated by some fundamental algorithms from this field.

2.1. Preliminaries

To keep notation relevant to this work as standardized and as simple as possible several variables and equations are introduced here. The notations mostly follow the ones from (Ross & Bagnell, 2014; Ross, Munoz, et al., 2011) as they are used in most literature reviewed in section 3 and follow the usual form for structured prediction and reinforcement learning. Notations introduced in the sections of the text are to be prioritized over these ones here.

- Source sentence of length I $X = \{x_1, ..., x_I\}$
- Target sentence of length J $Y = \{y_1, ..., y_J\}$
- Set of k actions $A = \{a_1, ..., a_k\}$
- Set of n states $S = \{s_1, ..., s_n\}$
- Current time step t with sequence length/ task horizon T
- $P_a(s, s')$ probability that action a in state s at time t will lead to state s' at time $t + 1$
- $R_a(s, s')$ immediate reward for transitioning from state s to state s' after taken action a
- Student's/learned policy π or $\hat{\pi}$ (depending on conditions)
- Expert's policy π^*
- Distribution of states d_π^t for policy π being executed from time step 1 to $t - 1$
- Expected cost $C(s, a)$ for performing action a in state s bounded by $[0, 1]$
- Expected cost $C_\pi(s) = E_{a\sim\pi(s)}[C(s, a)]$ according to π in s
- Total cost of executing policy π for T steps $J(\pi) = \sum_{t=1}^{T} E_{s\sim d_\pi^t}[C_\pi(s)]$
- Expected future cost-to-go $Q_t^\pi(s, a)$
- Cost-weighted training example (s, t, a, Q)

In this work mostly the finite horizon control problem in the form of Markov Decision Process (MDP)s with states s and actions a is considered. The i.i.d. assumption of the MDP can not be made in this setting because of the policy iteration in data and policy aggregation methods. Policies from previous generations are used to create demonstration that the current policy is learning of.
In the domain of machine translation it might not always be necessary to define an action a at time step t and state s. Considering each token x in a source sentence X is represented by an action a that leads from s at time step t to s' at time step $t+1$. In the decision process x_i is depended on previous the token x_{i-1}, therefore each token in a sequence X is unique, even if the token might be the same. Then it can be assumed that the graph that is created by the states and it's transitions/action is acyclic. Therefore for machine translation either the state dimension s or the time dimension t can be omitted because after taking a set of actions the possible actions in t and s will be equal $A_t == A_s$.
Furthermore the synonyms for student: learner and learned policy are used interchangeable and should be derived from the context. The same applies for synonyms of expert like teacher, oracle or expert policy.

2.2. Machine Translation

MT is the task of autonomously translating one language into another using a software translation system. On a basic level, the system tries to match a sentence in the source language to the best possible sentence in the target language.

2.2.1. Statistical Machine Translation

SMT is one type of MT and uses a data-driven approach where bilingual and monolingual corpora are used as main source of knowledge. The idea of SMT comes from the information theorem of (Shannon, 1948) where it is shown that the amount of information in a message can be represented as how surprising it is for the receiver. This is because generally speaking unknown information is more interesting than known information. The informational content I of an event decreases as the probability of the event occurring increases. (Shannon, 1948) describes the expected value of I as the entropy with a discrete random variable X as:

$$H(X) = E[I(X)] = E[-\log(P(X))] \tag{2.1}$$

Explicitly it can be written as:

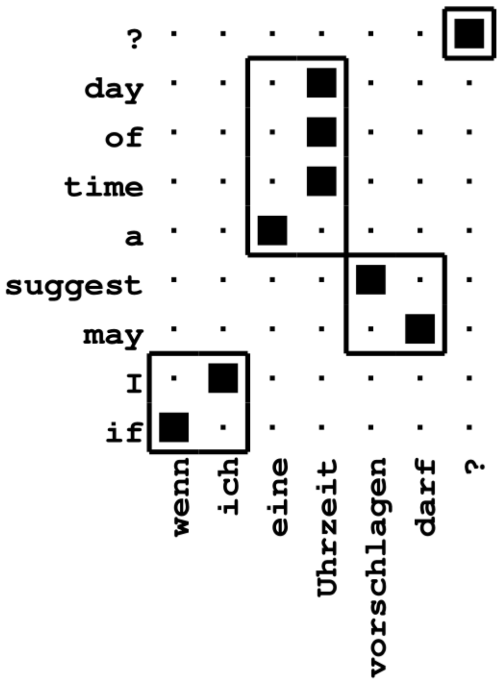
$$H(X) = -\sum_{i=1}^{n} P(x_i) \log P(x_i) \tag{2.2}$$

Where E is the expected value operator, n is the number of random variables and the logarithm is usually to the basis 2, 10 or e. This finding can be used in the task of MT as the source of information for a sentence can be represented as a sequence of symbols governed by a set of probabilities. It is shown that it will choose successive symbols

according to certain probabilities depending on preceding choices as well as the particular symbols in question. This is also know as a discrete Markov process, where each element in a finite sequence $\{S_1, S_2, ..., S_n\}$ has a set of transitions with probability $P_i(j)$ to go from state S_i to state S_j. To get information of the Markov process each transition corresponds to, depending on the model, a character, word or phrase. When going through the sequence of states the transition with the highest probability is taken, resulting in the most probable translation. For example in English it is more likely that character "H" is successive to "T" than to "Q" so the probability on the transition from state $P_H(T)$ is larger than the one of $P_H(Q)$.
Before in 2016 NMT systems became popular it was simply infeasible to train large scale neural networks. SMT systems were state-of-the-art and used by Microsoft, Google and Yandex (Translator, n.d.; Turovsky, 2016; Yandex Blog, 2017). SMT systems learn the correspondences between source and target language by training on coocurrences of usually words or phrases, that can be seen as distinct entities. Coocurrence tries to simulate semantic relations between entities and can be described as a higher level counting of them or a global representation of the context the entity occurs in. It is then possible to measure the statistical correlation between the coocurrences of such entities. They can also be seen as Mutual Information trying to measure the dependence and the information obtained from one entity through another (Bordag, 2008).
Because this approach does not rely on dictionaries or grammatical rules it provides the best translations using phrases where the context around a given word is used instead of trying to perform single word translations. The basic idea is to separate the source sentence into phrases, translate them into the target language and concatenate them in the correct order. To show why phrased-based translation are preferable in SMT systems an example from (Zens, 2008) is presented here. In figure 2.1 the German source sentence "Wenn ich eine Uhrzeit vorschlagen darf?" and the English target sentence "If I may suggest a time of day?" are separated into words and phrases. The black squares show the direct translation of German words (x-axis) and English words (y-axis). The boxes that envelop the squares represent the phrases that get matched from one language to another, the matching is also shown in the table on the right. Using the translation probability and Bayes rule the SMT system tries to find the best translation from source sentence X to target sentence Y as follows:

$$\arg\max_{Y} \; P(Y|X) = \arg\max_{Y} \; P(X|Y)P(Y) \tag{2.3}$$

By this a separate language $P(Y)$ and translation model $P(X|Y)$ are obtained. During decoding the input sentence X is split into phrases x_i. Each phrase x_i is then translated into a phrase of the target language y_i and might be reordered. The translation model $P(X|Y)$ is given by a probability distribution that is obtained by the statistical correlation of the coocurrences in the bilingual corpus and a distortion model that manages the different positions of phrases in the languages. Through n-gram overlap and usually an additional length penalty factor the language model $P(Y)$ is obtained. The best possible translation for X is then given by $Y_{best} = \arg\max_Y P(X|Y)P(Y)$.



source phrase	target phrase
wenn ich	if I
eine Uhrzeit	a time of day
vorschlagen darf	may suggest
?	?

Figure 2.1.: Example of phrase-based translation and the list of used phrase pairs (Zens, 2008).

SMT systems generally work well when trained on a large amount of parallel corpora. Parallel corpora are sentence aligned texts where each sentence is usually delimited by a break line. Each sentence from the source is matched exactly to one sentence of the target with the same id. Learning from coocurrences of phrases or words introduces problems when trying to translate into morphology-rich languages such as German. Often single instances of compound words in the target language occur during test time but have not been encountered during training. Additionally the system is mostly unable to correct typos or singleton errors. That is why SMT systems got mainly replaced by NMT systems.

2.2.2. Neural Machine Translation

SMT systems count phrase pairs and their occurrences to model similarities between them but they do not share statistical weights in the estimation of translation probabilities. On unseen data this leads to a general sparsity problem because it is unlikely that the context of distinct phrases are the same in different occurrences. NMT systems on the other hand are based on neural networks which share weights and can be an universal approximator for any function given the right architecture as proven by (Debao, 1993; Hornik, 1991) and others. For a classification problem, such as MT, usually the parameters of the network are trained using the cross-entropy. Entropy is the amount of bits required to transmit a randomly selected message from a probability distribution as shown in equation 2.2. When the target probability distribution is denoted as D and the approximation of the target probability distribution (the prediction of the model) as Q then the cross-entropy is the number of additional bits to represent an event using Q instead of D and can be written

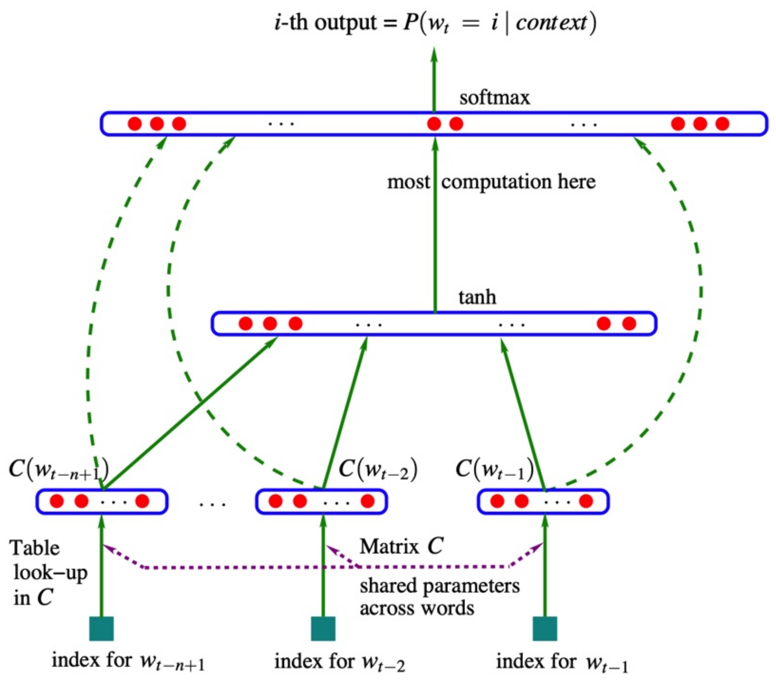


Figure 2.2.: Illustration of a feed-forward neural network for predicting a single output word at time t. (Y. Bengio et al., 2003)

as:

$$H(D,Q) = -\sum_{i=1}^{n} P_D(x_i) log P_Q(x_i) \tag{2.4}$$

Where n is the number of possible events x, $P_D(x_i)$ and $P_Q(x_i)$ are the probabilities of the event in distributions D and Q respectively.
Another function to discretely compare two distributions is the negative log-likelihood which is based on the maximum-likelihood estimation (MLE). For Bernoulli and Multinoulli probability distributions the cross-entropy and negative log-likelihood can be used interchangeably as a loss function and will yield the same result (Murphy, 2013). Before the output of the network for a multi class classification problem is used in the loss function they usually get passed to the softmax activation function. The softmax squashes the output of the network of size n into the range $0 - 1$ to get a normalized representation of the output values that sums to 1. It is defined as:

$$S(x_i) = \frac{e^{x_i}}{\sum_j^n e^{x_j}} \tag{2.5}$$

Where n is the number of classes and x_i is the network output for class i also called logit. We can then interpret the output of the softmax as the probabilities that a certain set of features belongs to a certain class.
(Y. Bengio et al., 2003) were the first to propose a feed forward neural network to learn the distributed representation for words to address the problem of data sparsity of SMT. Their concept for predicting a single word at time t by a sequence of input words using a neural network is shown in figure 2.2. They decompose the function of the neural network with a mapping C from any element word i in the vocabulary V to a real vector $C(i)$. Therefore

C represents the distributed feature vectors associated with each word in the vocabulary. C is represented by a $|V| \times m$ matrix of free parameters. For their experiments they used vocabulary size of $|V| = 17000$ and $m = 30, 60, 100$ features. Training their network for only 5 epochs on the AP news dataset with around 14 million words would take 3 weeks on 40 CPUs. At the time their model shifted the problem of MT to a computational problem of learning parameters, as they out-performed every state-of-the-art statistical or example based model by using learned representations of words. Due to the architecture of the network the model was still limited by a fixed context size, which later was solved by (Mikolov et al., 2010) who proposed a language model based on a Recurrent Neural Network (RNN) to obtain a dynamic context length. RNNs transfer information through time by connecting nodes from a directed graph along a temporal sequence. The context of previous predicted instances from time step $t-1$ in the sequence is given to the node at time step t. Therefore information can be transmitted through time and dependencies of words with respect to previous words can be modeled.

The next big improvement was done by (Kalchbrenner & Blunsom, 2013) who used an end-to-end encoder-decoder structure for machine translation. Their model encodes the sources sequence $\{x_1, ..., x_n\}$ to a new continuous representation $\{z_1, ..., z_n\}$ using a Convolutional Neural Network (CNN). In the decoder, z is then used to sequentially generate the target sequence $\{y_1, ..., y_m\}$.

Encoding a whole sentence into a single vector Z does not represent the context of each word but only the sentences as a whole. To overcome this problem (Bahdanau, Cho, et al., 2016) introduced the attention mechanism, which creates a representation of learned context vectors $\{h_1, ..., h_n\}$ for each z_n. At every time step of decoding, each context vector is of different importance and therefore the attention is set on different parts of the sentences representation for each decoding step. This allows for modeling dependencies in sequences without regard to their distance in the input or output sequences.

This mechanism was used and further improved by (Vaswani et al., 2017) who showed that convolutional and recurrent building blocks can be completely substituted with their self-attention mechanism. A simple example is given in figure 2.3, where for the input sequence *Thinking Machines* the first self-attention calculation is shown step by step. Each word of the sequence is embedded into a context vector x_i and multiplied by three weight matrices which are learned during training w^Q, w^K, w^V (not shown in the figure). The matrices correspond to queries, keys and values respectively, the resulting vectors are denoted q_i, k_i, v_i. Then a score is calculated by taking the dot-product of the query and the key vector and dividing it by the dimensionality of the key vector d_k [1]. The softmax function is then applied on all scores of the input and the respective values are multiplied to acquire a scaled value vector. These scaled vectors are then summed to get the output of the self-attention for the first word.

To extend the model's ability to focus on different position in the sequence they introduce multi-head-self-attention. The method creates multiple self attention heads which all have their own weight matrices w_n^Q, w_n^K, w_n^V, where the number of heads n is usually 4 - 8.

[1]According to (Vaswani et al., 2017) the $\frac{1}{\sqrt{d_k}}$ factor is applied to obtain more stable gradients.

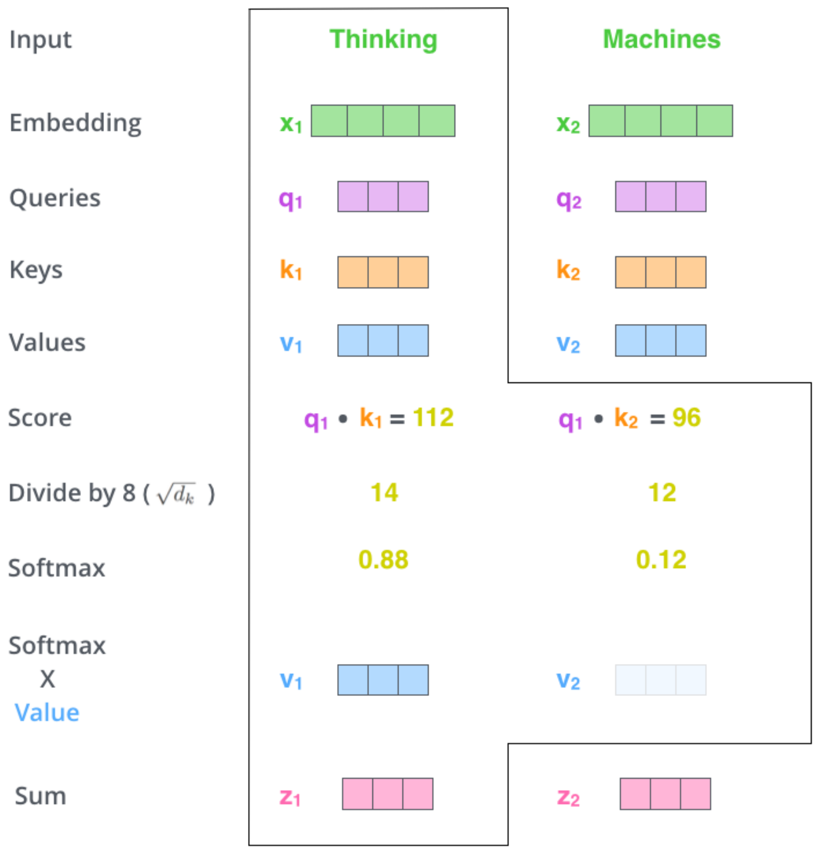


Figure 2.3.: Overview of self-attention calculation for a single instance of a simple sequence. (Alammar, 2018)

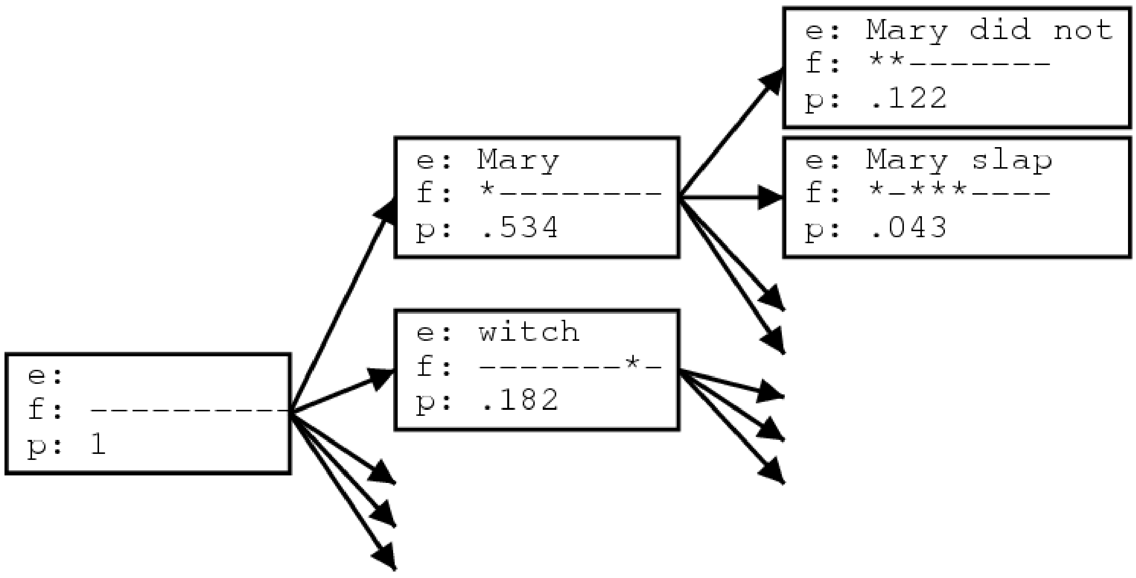


Figure 2.4.: Moses decoder using the beam search algorithm (Koehn, 2013)

Thus the calculation for each word needs to be done n times. Additionally the output z for each attention head is scaled by another learned weight matrix, to scale the representation subspaces of each individual head.
Their Transformer architecture is completely based on self-attention in the encoder and the decoder and set new state-of-the-art performance in less training time, because it is not bound to the sequential part of the decoder and therefore it is more parallelizable. To this day, most competitive models for MT are still based on the Transformer architecture.

2.3. Machine Translation Utilities

In this section utilities that are used later in the course of this work are shown. For instance the SMT toolkit Moses will be reviewed in more detail as it provides the expert for the imitation learning with knowledge. Furthermore the most common metric for the evaluation of MT systems will be introduced. Additional some metrics that are closer to human judgement are shown.

2.3.1. Moses SMT

A popular open source toolkit for phrase-based SMT is called Moses by (Koehn et al., 2007). It can train language models for two language pairs only using sentence parallel corpora. During training Moses learns the coocurrences between two languages and exports the phrases together with their statistical correlation into lookup-tables, also called phrase-tables. To accelerate decoding, possible phrases for the translations of the source sentence are picked from the phrase-tables before the decoding begins. In figure 2.4 an example of the beam search of Moses is shown. Each box in the graph has three rows the first one marked with **e:** represents the current target translation hypothesis, row **f:** labels the words in the source sentence that are already translated with an asterisk and the last row **p:** shows the current accumulated probability. During decoding the target output

is generated from left to right in phrase sized steps. New states are sequentially created by extending each hypothesis with phrases of translations that covers some of the source input words which were not yet translated. The current probability of the new state is the probability of the original state multiplied with the translation, distortion and language model probability of the added phrasal translation, following the model described in section 2.2.1. Final states are hypotheses that cover all target words. Among these the hypothesis with the highest probability is selected as best translation.
Through this exhaustive search all possible translations are obtained. To optimize the algorithm hypothesis that never have the chance of being among the best translations are being discarded early. When considering all possible hypothesis (states), the upper bound for the number states can be estimated as: $N \sim 2^{n_s}|V_Y|^2 n_s$. Where n_s is the number of words in the source sentence and $|V_Y|$ is the size of the target vocabulary, thus 2^{n_s} becomes the critical part and needs to be limited. In practice the number of sensible target words $|V_Y|$ gets lower for each word generated, as in most cases repeating words for examples should have a very low probability. Each state that is used in the beam search by Moses not only includes the cost $(1 - probability)$ of the currently build hypothesis but also the future cost that estimates how much it will cost to continue the sentence from this point. The future cost estimation favors hypotheses that already covered difficult parts of the sentence and discount hypotheses that covered the easy parts first. To limit the search space (lattice) these metrics are used in two different pruning methods. Histogram pruning keeps a fixed number of hypothesis at any time, while beam threshold pruning limits the number of hypothesis based on a relative factor α of the probability of the best hypothesis P_b. So every hypothesis that has a probability $P_i < \alpha P_b$ gets pruned. For the expert described in this work it is beneficial to use threshold pruning. Whenever pruning is mentioned in the context of Moses search spaces, it can be assumed that threshold pruning is meant (Koehn, 2013).
In this project Moses is used to generate search spaces for each source sentence and saving them as files. This provides knowledge of the target sentence through the learning process of Moses and is used in the oracle, the teacher for the reinforcement learning algorithm, later.

2.3.2. BLEU

Human evaluations of machine translation is the best method to qualitatively compare translations, but it is also expensive and time consuming when comparing large-scale data sets. Thus the automatic evaluation metric BLEU was proposed by (Papineni et al., 2002) and it is still the pseudo standard when evaluating translations today. To obtain near human evaluation accuracy BLEU uses a modified n-gram precision that clips each n-gram to the maximum number of times it occurs in the reference. For example:

- Candidate: the cat sits on the tree
- Reference: a cat is sitting on the tree

All candidate n-gram counts and their corresponding maximum reference counts are collected. The candidate counts are clipped by their corresponding reference maximum value.

In this example there are five 1-grams, two 2-grams and one 3-gram. The word “the” occurs twice in the candidate but only once in the reference, thus the 1-grams are clipped to a count of four. The candidate counts are then summed and divided by the total number of candidate n-grams with the following equations from (Papineni et al., 2002):

$$BLEU = BP \cdot exp(\sum_{n=1}^{N} w_n \log p_n) \tag{2.6}$$

Where N defines the maximum n-gram size, w_n is an uniform weight $1/N$, p_n is the modified precision as described above and the brevity penalty BP is defined as:

$$BP = \begin{cases} 1, & \text{if } c > r \\ e^{1-r/c}, & \text{if } c \leq r \end{cases} \tag{2.7}$$

Here r and c are the number of words in the reference and candidate respectively. In a more abstract way 1-grams are used to measure the equality, where n-grams of further order are used to compare the fluency of candidate and reference. Throughout this work BLEU is used in many different places showing it’s capabilities in measuring MT performance but also demonstrating it’s drawbacks.

2.3.3. Metrics Closer to Human Judgement

In this section three additional metrics for natural generated language that are closer to real human judgement will be introduced briefly.
Firstly METEOR (Metric for Evaluation of Translation with Explicit ORdering) by (Banerjee & Lavie, 2005) was designed to fix some of the problems found in BLEU, and also produce good correlation with human judgement at the sentence or segment level. It is based on the harmonic mean and recall of the unigram precision. On a corpus level it correlates up to 96.4% to human judgement, while BLEU only achieves 81.7% (Banerjee & Lavie, 2005).
BLEURT by (Sellam et al., 2020) indicates to what extent the candidate conveys the meaning of the reference. Like METEOR is used to bring evaluation closer to human judgement. Models are not calibrated like BLEU as the results are not in the range of $[0, 1]$. Instead, they mimic the WMT Metrics Shared Task’s human scores, which are standardized per annotator. It is suggested to interpret the absolute values rather than using the metrics for comparison. BLEURT creates these ratings by pretraining BERT (Bidirectional Encoder Representations from Transformers) from (Devlin et al., 2019) for quality evaluation. Generated text from the MT system is again evaluated by the BERT model which has access to multi dimensional representations of used words. In contrast BLEU is limited to the n-gram overlap and there can not evaluate sentences in the same way. BLEURT is particularly robust to both domain and quality drifts and showed that it can model human assessment with superior accuracy.
The GLEU score is an evaluation metric by (Mutton et al., 2007) that measures fluency by examining different parser. Each produces output that can be taken as representing

degree of ill-formed grammar. In their paper it is shown that is correlates more to a human machine translation rating than for example BLEU.

2.4. Learning to make decisions

In this section different learning methods will be reviewed. The problem definition of taking independent decisions in every state that maximize the goal of the task will be shown. Therefore equations and theorems will be introduced that are needed to describe the problem. The main focus will be on learning how to take decisions with RL, IL and Inverse Reinforcement Learning (IRL).

2.4.1. Introduction to Reinforcement Learning

In supervised learning the learner makes a prediction based on the current input and the learned parameter. It then gets a feedback from a loss function that defines the difference of the prediction and the references given the current input. The parameters are modified by their total contribution to the loss. The objective for the system is to generalize it's responses so that it does good predictions on unseen data.
Unsupervised learning generally addresses problems where the goal is to find structures or clusters in the data without references or supervision.
In reinforcement learning there is no direct signal for every prediction (action) that shows how good or bad that action was - the learner is not directly told which actions to take. Instead there are rewards that are given at any point in the sequence of actions by a reward function. Rewards might be delayed, such that an action that was taken in the beginning of the sequence yields a reward at the end. To discover the actions that maximize the reward, the learner has to try non-optimal local actions and therefore get information about the possible global reward (Sutton & Barto, 2018).
In general it is the process of making a sequence of decisions which goal it is to maximize the cumulative reward. These decision processes are based on the Hidden Markov Model (HMM) where each state possesses the Markov property. The Markov property says that a future state s_{t+1} with $t \in \{1, ..., N\}$ is only dependent on the current state s_t where S is a finite set of N states. Thus state is a Markov state if:

$$P(s_{t+1}|s_t) = P(s_{t+1}|s_1, ..., s_t) \tag{2.8}$$

Generally the HMM can then be described as:

$$P(s_t, a_{1:t}, o_{1:t}) = P(o_t|s_t) \int P(s_t|s_{t-1}, a_t) P(s_{t-1}, a_{1:t-1}, o_{1:t-1}) ds_{t-1} \; P(a_t) \tag{2.9}$$

Where o_t, a_t and s_t are the observation, action and state at time t respectively. The parts of the equation have different responsibilities regarding observations, motions and actions. For example, an automatic cleaning robot is used to clean the floor of a room. It's task is to clean all possible positions (states) of the floor and do movements (actions) based on measurements and it's current location. The sensors for the measurements and the electric motors for the motions are not perfect and have tolerances, thus their uncertainty needs to

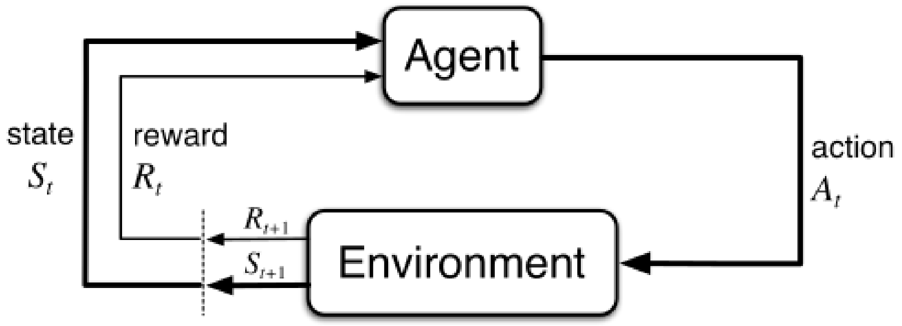


Figure 2.5.: The agent–environment interaction in a Markov decision process (Sutton & Barto, 2018).

be taken into consideration. If for example the motor only has 95% probability of taking an action it needs to be modeled probabilistic. Therefore the measurement model is defined as the probability of observations depending on the current location $P(o_t|s_t)$. The motion model is given by the probability of the current location conditioned on previous location and the current action $P(s_t|s_{t-1}, a_t)$. The current action is also modeled probabilistic with $P(a_t)$. Now assume the robot has left the station and did a couple of actions. Then $P(s_{t-1}, a_{1:t-1}, o_{1:t-1})$ is the old belief of the last location s_{t-1}. Based on the old belief, the motion model, the measurement model and the current action a current belief is estimated $P(s_t, a_{1:t}, o_{1:t})$.
In situations where modeling of decision making is required usually an extension of HMMs are used - Markov Decision Processes (MDPs). MDPs add a reward function to the HMM which is based on the action a_t taken in state s_t and describes how good or bad an action in a certain state was. The reward is defined as $R(s_t, a_t)$ with the state transition probability $P_{sa}(\cdot)$. In figure 2.5 the agent-environment interaction in a MDP is shown. The agent or policy π maps states to actions $S \rightarrow A$ in discrete time steps; actions interact with the environment; the agent is given a reward and new state by the environment after each action. A MDP can then be defined as a tuple of $(S, A, P_{sa}(\cdot), \gamma, R)$ where R is the reward function and γ the discount factor for rewards. The agent's objective is to maximize the amount of reward it receives over time. By obtaining rewards after each action, be it immediate or delayed, RL can solve MDP without explicit specification of the transition probabilities. Generally taking an action and not receiving an immediate reward is unintuitive, as the agent expects a reward or cost based on the last action. To overcome this issue an estimation of a value function is introduced that estimates how good it is for the agent to be in a given state. Where "how good" is defined by the expected future reward for being in that state. Accordingly value functions are based on the policy which describes the way of acting in states. A value function for policy π evaluated at state s_t is given by:

$$V^{\pi}(s_t) = E[R(s_t) + \gamma V^{\pi}(s_{t+1})|\pi] \tag{2.10}$$

Where the value V is the expected value over the distribution of the state sequence $(s_t, s_{t+1}, s_{t+2}, ...)$ that is gone over when executing π starting from s_t. Furthermore an action-value function can be defined as:

$$Q^{\pi}(s_t, a_t) = E[R(s_t, a_t) + \gamma Q^{\pi}(s_{t+1}, a_{t+1})|\pi] \tag{2.11}$$

An optimal value or action-value function is then given by the policy that maximizes all values V or action-values Q.

$$V^*(s) = \max_{\pi} V_{\pi}(s) \tag{2.12}$$

$$Q^*(s, a) = \max_{\pi} Q_{\pi}(s, a) \tag{2.13}$$

Additionally with the Bellman optimality it is know that there must be a deterministic optimal policy for the MDP, such that π is then an optimal policy iff for all $s \in S$.

$$\pi(s) = \arg\max_{a \in A} Q^{\pi}(s, a) \tag{2.14}$$

The goal of RL is to find a policy π that maximizes the value function for all $s \in S$ as shown by (Sutton & Barto, 2018). In many real world applications RL has shown it's potential. For example in the early 90's the TD-Gammon of (Tesauro, 1992) learned to play back-gammon on a grandmaster level with very little background knowledge on the game. There are many more application such as optimizing DRAM access (Ipek et al., 2008), mastering the complex game of GO (Silver et al., 2016) and more.
Through reinforcement learning it has been shown that taking near optimal actions in complex unknown environments can be learned. Trying every possible action in such an environment and observing the reward can be costly, thus computational intensive. Learning from demonstrations before exploring the environment has shown to be more effective and will be reviewed in the next sections.

2.4.2. Imitation and Inverse Reinforcement Learning

RL works well when the reward function, or at least a good approximation of it, is known for most pairs of actions and states. In some tasks defining a reward function might be difficult due to the complexity of the environment. To give an example from a natural perspective: For a human the most general reward signal for any kind of task could be the release of dopamine. For a student this can be achieved by getting a good grade after studying the whole month or getting a paycheck after a long day of work. For young kids this is not that trivial since they do not know what is right or wrong; they do not know how to interact with the environment such that a reward is released. Just through exploring their surrounding environment the learning process would be very slow. That is why they usually try to learn from the behavior of their parents **and** by exploring.
In machine learning this concept is called imitation learning, apprentice ship learning or learning from demonstrations. In the simplest form this can be done by directly trying to copy it. This is called behavioral cloning and generally is closer to supervised learning where the student learns a policy which maps states to actions directly. The student is usually defined by a set of parameters, such as a neural network, which are fit to solve a

certain problem. This way of learning was popular in the early stages of imitation learning. Examples which successfully applied this method with experts' demonstrations as training data are (Hayes & Demiris, 1995; Isaac & Sammut, 2003; Pomerleau, 1989).
To give an example where behavioral cloning does not work: A student (policy being trained) should learn how to drive a car from point A to point B within a city. The student already knows how to drive in general but does not know the city very well. The expert (who knows the city) can recommend a path from point A to B. At each intersection (state) along the path the expert gives an indication (action) where to go. Assuming the student learned from the expert's behavior and follows the expert's path the first few steps but eventually takes a wrong turn. The student will have to turn around until the path of the expert is reached again. It will only get to point B, it's destination (and obtain a reward), when following the expert's path. Therefore the student is forced to clone the behavior of the expert.
This can be improved by Direct Policy Learning (DPL) which uses an interactive expert that can finish a path or trajectory at any given state. The method works as follows:
During training the student predicts a trajectory $S_{1:T}$ by rolling out it's policy to a certain state s_t at time step t. Where T is the number of states in the complete trajectory. The expert is then queried to continue the trajectory from s_t and maximize the expected future reward. s_t is either chosen randomly or at a point where the student is uncertain what action to take. In the scenario of the example the reward is only obtained at the end of the sequence. When using behavioral cloning for training the student, this means there would not be a signal how good the trajectory $S_{1:t-1}$ is. There is only a 0-1 reward signal how good the action a_t in state s_t was. Thus the student would only learn to take the expert's action a_t^* in state s_t. In DPL on the other hand, the student learns to recover from it's mistakes with the expert's continuation $S_{t:T}$.

In the context of MT usually the Maximum Likelihood Estimation (MLE) is used as a multi-class classification objective. Unfortunately, this training objective is not always a good surrogate for the test error because it only maximizes the ground truth probability and ignores the wealth of information provided by structured losses. Furthermore, it introduces inconsistencies between training and prediction (such as exposure bias), which may negatively impact test results. In IL a learner tries to mimic an expert by observing it's behavior. Based on the observed the learner tries to map states to actions. Rewards give the learner a signal of how good the last action in that state was. Delayed rewards lead to more complex decision making, as the learner might not immediately receive an useful signal. Cloning the expert's behavior works for learning direct mappings but is insufficient for more complex reward functions. Imitating an expert can usually not be described as a straight forward prediction. Learning a hidden reward function in IL is based on matching features of states from the learned to the expert's policy. Decisions are made using structured predictions which have multiple dependencies between them. For example when translating a sentence with a MT system, each word has strong conditional dependence on the sequence of words before. Generally a structured prediction problem can be defined as a normal supervised learning problem:

$$L = E_{(x,y)\sim\mathcal{D}}\left[\ell\left(y^*, \hat{y}; x\right)\right] \tag{2.15}$$

Where $\mathcal{D}$ is an unknown distribution of input space x and output space y. The loss ℓ defines the distance between the true value y^* and the predicted value $\hat{y}$ given x. Therefore doing structured prediction in a MDP can be described as matching of features for each individual decision. More specifically in IL the knowledge is transferred from features of decisions from the expert to the student via structured predictions.

(Abbeel & Ng, 2004) proposed that approximating the expert's reward function directly is more desirable. This is because there is generally a difference in learning how to copy expert's demonstrations and learning the hidden objectives of the expert that maximize the reward. The problem of deriving a reward function from observed behavior is called inverse reinforcement learning. Unlike RL where the reward is obtained at the end of each sequence, the goal of IRL is to learn rewards for each action of the expert. (Ng & Russell, 2000)
The goal of IRL is to find a set of possible reward functions $R(s)$ such that the expert's policy π^* is the optimal policy π in the MDP. For simplicity then assume that optimal policy π is $\pi(s) \equiv a_1, \forall s$. From the Bellman optimality it can be said that $\pi(s) \equiv a_1, \forall s$ iff

$$(P_{a_1} - P_a)(I - \gamma P_{a1})^{-1} R \succeq 0 \tag{2.16}$$

Where P_{a_1} and P_a are probability matrices of taking the optimal action a_1 and the every other action a. $\succeq$ denotes that every element in the matrix or vector should fulfill the condition of $\geq$. Therefore for every action taken the reward R needs to be ≥ 0. For a small set of states this can be solved with linear programming.
For the case of infinite state space (Ng & Russell, 2000) did the assumption that there is a finite set of feature functions $\phi_i(s)$ for approximating the value function. Therefore rewards can be defined as a sequence of linear functions of features:

$$R(s) = \alpha_1\phi_1(s) + \alpha_2\phi_2(s) + ... + \alpha_d\phi_d(s) \tag{2.17}$$

Where ϕ are feature mapping functions that project from the state space into real numbers and α are the parameter to be learned. This means that a policy can be learned that is as good as the expert by matching feature function ϕ by:

$$\mu(\pi) = E_\pi[\sum_{t=1}^{\infty} \gamma^t \phi(s_t)] \in R^k \tag{2.18}$$

$$V^\pi(s) = w \cdot \mu(\pi) \tag{2.19}$$

(Ratliff et al., 2006) used this feature mapping for learning behaviors as a maximum margin structured prediction problem over a space of policies. Such that an optimal policy can be trained on the structure prediction problem to mimic the expert's behavior. Their formulation is based on Support Vector Machine (SVM) objective and can be written as:

$$R(w) = \frac{1}{N}\sum_{i=1}^{N}(\max_{y\in Y}(w^T\phi(x_t, y) + \ell(y_t, y))w^T\phi(x_t, y_t)) + \frac{\lambda}{2}||w||^2 \tag{2.20}$$

Through this the deviations from the expert's demonstrations are penalized and the learner is driven to produce similar outputs as the expert. They validate their results by planning trajectories on satellite images of paths, such that a certain cost is reduced. The two objectives that they observe are staying on the road along the path and hiding in the trees (staying on the road as short as possible). For both objectives they obtain good results, representing near optimal paths. Another example where IRL was applied successfully is the work of (Shimosaka et al., 2014). They showed that learning rewards from the expert's behavior can be used in complex real world scenarios. In their work they proposed a formulation of defensive driving on residential roads with IRL. The authors provide new feature descriptors for computing reward functions to represent risk factors on residential roads such as corners, barrier lines, and speed limitations. Their results show that they successfully managed to model risk anticipation and defensive driving for this scenario.
Throughout this section different imitation and inverse reinforcement learning methods were reviewed. All of these methods try to solve MDPs, but there is no method that works well on every decision process. The method that should be used for a specific problem is depend on many aspects.

- Is a reward function given or can it be easily approximated?
- What complexity does the environment have?
- Are there expert's observation that can be used during training?
- Can the expert be queried interactively?

These questions should be answered before defining the learning method that solves the MDP representing the global problem.

2.4.3. Fundamental Algorithms

In this section different methods for solving tasks that are modeled with MDPs will be reviewed. As stated in section 2.4.2 MDPs describe decision processes where each state is assumed to have the Markov property, so that it is only depended on the current state and action. They express the framework for learning to take decisions in known and unknown environments. Over the last years many problems were solved using these methods: in games, path finding and other structured prediction tasks. Here a theoretical view on some of the algorithms used in these tasks is given.

(Daumé et al., 2009) proposed a method for Search-based Structured Prediction (SEARN) which decomposes structured prediction into multiple classification tasks, where each label of the structured output is predicted by a classifier. The global task is then to optimize the unknown reward/loss function that can be expressed as a linear combination of these

classification tasks (features). In general the student generates a roll-in (trajectory) consisting of states and actions till the end of the sequence T. The expert creates for each action a_t in the sequence a roll-out with the expected future cost estimate. Such estimates place the expert back in an identical state for each action and require T roll-outs. SEARN iteratively improves the learner by generating a new non-stationary policy every epoch. At the end of each epoch the weights of the policies from the previous epochs are aggregated to obtain a new policy. This policy is then used for the structured prediction in the new epoch.
Similarly SMILe by (Ross & Bagnell, 2010) takes the approach of SEARN but reuses already observed errors, such that the total number of required roll-outs decreases. SMILe can also be adapted for general structured prediction and would provide similar guarantees to SEARN but without requiring roll-outs of full expected future cost for every prediction. These kinds of methods are know as policy aggregation methods and output a stochastic learned policy.
In contrast, DAGGER, proposed by (Ross, Munoz, et al., 2011) aggregates data each epoch to improve the same learner. The learned policy is therefore stationary and deterministic and in their work they show that the policy is guaranteed to perform well after a certain amount of made errors. Polices trained with SEARN or SMILe do not have strong guarantees and can fail in real world scenarios due to stochastic nature (Ross, Munoz, et al., 2011). To show the difference of policy and data aggregation the training routines of DAGGER and SEARN are shown in algorithm 1. Both algorithms expect a training dataset X for the supervised learning part, an expert's policy π^* and a learning rate β. Unique lines of the code from the respective methods are labeled with *#SEARN* and *#DAGGER*, the rest of the code remains the same for both.

At first a probability p is defined by the learning rate β and the current epoch i (line 3). A stochastic policy π is then sampled using p, such that for each action the expert's policy π^* is used with a probability p or the learned policy from the last epoch H_{i-1} is used with a probability $1-p$ (line 4). π is used to predict each instance of the training data. For each instance x and each action $\hat{a}_t$ a cost-sensitive tuple is generated (line 9-16). Representative features Φ_t are extracted given x and all previous actions $\hat{a}_{1:t-1}$ (line 9). Depending on the method either the stochastic policy π or the expert's policy π^* is used to predict, for each possible action a_t^j in x, the continuation $a'_{t+1:T}$ that minimizes the expected future cost. Then based on the previous actions $\hat{a}_{1:t-1}$, the current action a_t^j and the predicted continuation $a'_{t+1:T}$ the loss w.r.t. the ground truth is calculated (line 13). A new tuple of features and cost is created for every possible action in $\hat{a}_{1:T}$ and added to D. At the end of each epoch a new policy h_i is trained on D (line 18).
The difference of DAGGER and SEARN for collecting data is in line 5, where SEARN sets the dataset of examples to $\emptyset$ every epoch, DAGGER concatenates all examples. For DAGGER the knowledge of previous policies is represented by the aggregated data and transferred to the new policy by training on this data. Where for SEARN all previous policies are summed and the learning rate is used as an additional scaling factor to transfer more knowledge of more recent policies to the new policy. This way both methods become Follow-The-Leader (FTL) algorithms, where in each iteration the knowledge derived from

Algorithm 1 Training routine for SEARN and DAGGER

```
Input: training dataset $X$, expert's policy $\pi^*$, learning rate $\beta$
Output: best policy on validation set $H_n$
 1: Dataset of examples $D = \emptyset$
 2: for i to n do
 3:     $p = (1-\beta)^{i-1}$
 4:     $\pi = p\pi^* + (1-p)H_{i-1}$
 5:     $D = \emptyset$ #SEARN
 6:     for x in X do
 7:         Predict $\pi(x) = \hat{a}_{1:T}$
 8:         for $\hat{a}_t$ in $\pi(x)$ do
 9:             Extract features $\Phi_t = f(x, \hat{a}_{1:t-1})$
10:             for all actions $a_t^j$ in $\hat{a}_t$ do
11:                 Predict $a'_{t+1:T} = \pi(x|\hat{a}_{1:t-1}, a_t^j)$ #SEARN
12:                 Predict $a'_{t+1:T} = \pi^*(x|\hat{a}_{1:t-1}, a_t^j)$ #DAGGER
13:                 Estimate $c_t^j = \ell(\hat{a}_{1:t-1}, a_t^j, a'_{t+1:T})$
14:             end for
15:             Add $(\Phi_t, c_t)$ to $D$
16:         end for
17:     end for
18:     Train policy $h_i$ with $CSC(D)$
19:     $H_i = \beta \sum_{j=1}^{i} \frac{(1-\beta)^{i-j}}{1-(1-\beta)^i} h_j$ #SEARN
20:     $H_i = h_i$ #DAGGER
21: end for
```

the expert through cost-sensitive examples is increasing over the number of epochs.

Another method for approximating a global reward function from an expert using data aggregation is called AGGREVATE. It was proposed by (Ross & Bagnell, 2014) and is an extension to DAGGER that learns to choose actions to minimize the cost-to-go of the expert rather than the 0-1 classification loss of cloning it's actions. The major difference between the methods is that for only one so called exploration action a_t in $\pi(x)$ a continuation from the expert is done (as in line 12 of algorithm 1). The position t in the sequence $\pi(x)$ is thereby sampled uniformly from $t \in \{1, 2, ..., T\}$. In their paper they define the cost-to-go of the expert's policy π^* from t till T as: $\hat{Q} = \ell(a'_{t:T})$. Therefore $\hat{Q}$ can be seen as a replacement for cost c_t^j in line 13. Less formally, it represent the expert's opinion of how to continue a sentence optimal from a (non-optimal) starting point given by the student. The cost-weighted examples $\{s, t, a, \hat{Q}\}$ are then added to D as in line 16. Like in DAGGER a policy with a Cost-Sensitive-Classifier (CSC) is trained on all examples that are collected during the epochs.
One difference of the two methods is the amount of examples collected for each sample x in X. Where DAGGER observes an action a_t^j chosen by the expert in every visited state t along the trajectory and therefore collecting T data points, AGGREVATE only collects a single one per trajectory.
This work follows the concept of AGGREVATE because querying an expert for every state in the trajectory is not feasible due to it's prediction speed. The algorithm used will be reviewed in more detail in chapter 4.3. Additionally the amount of information or entropy

obtained by each example generated by the expert along the trajectory decreases due to repetitions of continuations. To give an example from the MT domain when translating from German to English:

- Source: Er spielt mit dem Ball im Hof
- Reference: He is playing with the ball in the yard
- Student's translation: He plays with the ball at home

Then each of the expert's continuation with it's respective cost-to-go estimates could be something like this:

1. He - is playing with the ball in the yard (0.0)
2. He plays - with the ball in the yard (0.2)
3. He plays with - the ball in the yard (0.2)
4. He plays with the - ball in the yard (0.2)
5. He plays with the ball - in the yard (0.2)
6. He plays with the ball at - the yard (0.4)

In this example the student took two actions differently than the expert. Translating "spielt" with "plays" and "Hof" with "home". In either method the student is learning from it's mistakes through the expert's future cost estimates. Learning on the examples above will improve the student but is costly and inefficient. As the amount of additional entropy in the examples from 3-5 is insignificant because the cost related to the mistake made in "plays" is already included in example 2. Additionally in the worst case the number of examples generated in each epoch can be up to $T \cdot X$. Where T here is the average number of tokens in a sequence and X the size of the dataset. This makes training a MT system on a large dataset using DAGGER impractical. AGGREVATE will sample a random exploration action for each instance x in the training data, thus creates X new samples every epoch. Generally the information obtained from examples generated in epoch D_i will be lower than the ones generated in epoch D_{i+1}, such that the total amount of information gained from each sentence when training with AGGREVATE is higher than with DAGGER.

An improvement to SEARN was proposed by (Chang et al., 2015). With their approach LOLS they show how to train a policy that learns how to search better than the reference policy/teacher. Their method follows the concept of SEARN but only uses the learned policy from the previous epoch as a roll-in policy and a stochastic policy for roll-outs. To repeat, SEARN uses a stochastic policy for roll-ins based on β. In their analysis they observe the cost of generating examples with deterministic and stochastic policies for both roll-in and roll-outs. By taking one-step deviations and executing different actions a_t^j for all t in T they try to find local optimal decisions even when the roll-in policy is sub-optimal. They show that using the reference policy as roll-in causes the state distribution to be unrealistically good. As a result, the learned policy never learns to correct for previous mistakes, performing poorly when testing. The concept of LOLS policy mixture was adapted in ADBLEU and will be shown in section 4.3.

3. Related Work

This chapter will be mainly focused around similarities and differences of other work related to this thesis. Analogies and comparisons will be made for methods that have objectives similar to the method proposed here.

For instance, (Zhang et al., 2017) proposed a method to combine the advantages of traditional SMT and NMT by exploiting an existing phrase-based SMT model. Contrary, the SMT system is used to calculate a phrase-based decoding cost for the n-best outputs from NMT system. Based on the cost a ranking is generated. The best translation according to the ranking is passed back to train the NMT system. The challenge is that the NMT outputs may not be in the search space of the standard phrase-based decoding algorithm. For the interface between NMT and SMT they propose a soft forced decoding algorithm which tries to find the best path in the SMT search space given the NMT input. They show that the re-ranking of NMT hypothesis with a SMT system can improve the test error of the NMT system.

A different approach was proposed by (Sun, Venkatraman, et al., 2017) who derived a policy gradient method for imitation learning and sequential prediction. They are using the reduction of IL and sequential prediction to online learning as shown in (Ross & Bagnell, 2014) to learn policies represented by expressive differentiable function approximators. For that they compare IL to RL using cumulative regret $R_N = \sum_{n=1}^{N} \left(\mu\left(\pi_n\right) - \mu\left(\pi^*\right)\right)$ to measure the speed of learning. Furthermore they validate their theory on robotics simulations and dependency parsing on handwritten algebra. For every test they did the trained policy reached near expert or sometimes even super-expert levels of performance. Their results show that IL is significantly more effective than RL for sequential prediction with near optimal cost-to-go oracles.

Closely related to this (Leblond et al., 2018) proposed SEARNN, a method that learns to search like SEARN but models each state of the MDP with a cell of a RNN which is also used as a differentiable function approximator. Like SEARN they try to minimize the difference between the global and the local objective by learning to search for the trajectory with the least cost. Their training method follows the one shown in algorithm 1 from section 2.4.3 with some modifications.
For training they use two different policies for roll-in and roll-out. The roll-in is defined as a context of all actions taken $a \in \{a_1, ..., a_{t-1}\}$ by the roll-in policy. This can be seen as the feature extraction of the RNN, where the hidden state h_{t-1} yields the features used in the roll-outs. The roll-out RNN is used to create a cost-sensitive example for each possible action a_t for time step t in the sequence. The resulting cost vector for each time step t is used in the loss calculation for each RNN cell. An intermediate dataset from tuples of initial state, all previous actions and cost vector is created. For the classification the RNN itself is used.

In figure 3.1 an example of the roll-in/roll-out mechanism is shown in the context of OCR, where the goal is to predict the characters in the image correctly. The roll-in

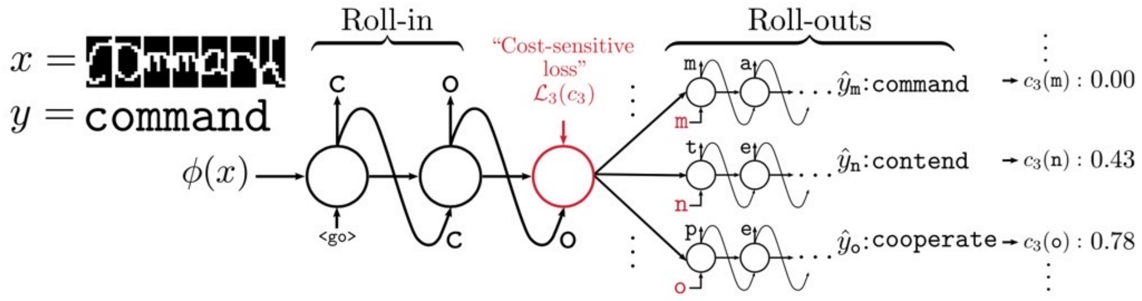


Figure 3.1.: Illustration of the roll-in/roll-out mechanism used in SEARNN for OCR from the paper of (Leblond et al., 2018).

extracts features sequentially by passing the context of the characters of each RNN cell to the next. Here for $t = 3$ each possible character/token is taken to observe the cost estimate from the roll-out RNN. This is done for every token in the set of token V and for every time step in T, thus a cost matrix with the size $V \times T$ is obtained.
SEARNN could also be applied to use an external oracle by using teacher forcing. During teacher forcing parts of the reference are given to the network as input, such that these tokens have to be correct. Using parts of the expert's output during teacher forcing would yield additional references. This way the student would generalize better on unseen data. To amplify the effect SEARNN could be also combined with DAGGER to create T references for each training instance.
Analog to this (S. Bengio et al., 2015) proposed to bridge the gap between training and testing time with scheduled sampling. During training their approach is to use scheduled sampling of either the reference or the model itself to generate the previous token a_{t-1} for the token under consideration a_t. In the early phases of training scheduling was set, such that a_{t-1} is sampled from the reference. The more the model approaches convergence the more a_{t-1} is sampled from the model itself. Their objective was to improve the performance of RNNs in the supervised learning context. Still it can be applied to imitation learning, where the scheduling is based on the expert and the learned policy. Like DAGGER the method requires multiple queries to the expert in one training sample and therefore can be too slow to use during training time.

(Cheng et al., 2018) proposed LOKI, a strategy for policy learning that combines IL and a policy gradient RL method. Their method executes an IL algorithm for a random number of iterations and then switches to RL. They show that if the time of switching is set correctly the learner can outperform the expert. The idea is to use the supervised aspect of IL in the early stages to quickly bring the initial policy closer to the expert's policy and then use the exploration of RL in the later to improve performance.
Another method that also tries to bridge the gap and interpolates between imitation learning and reinforcement learning is THOR by (Sun, Bagnell, et al., 2018).

The methods reviewed in this chapter try to train a policy by mimicking an expert. Different procedures are applied to reduce the domain gap between training where a reference or expert is available and testing where it is absent. Most of them try to tackle the problem by using stochastic policies of student and expert to generate roll-ins and roll-outs. By

using the student's policy for roll-in the model is trained to recover and learn from it's mistakes. Taking differences of estimated future cost-to-go for complete trajectories instead of a 0/1 loss for each action, help the student by understanding mistakes better and therefore improving faster. These methods mainly differ in the way that they transfer the knowledge from the expert to the student, either by doing different types of aggregation or changing the mixture of roll-in and roll-out policies.
In contrast, this thesis shows if it is possible to directly learn the global objective through IRL. It is therefore novel that it uses this approach in the topic of MT to directly optimize for the BLEU score. In the paper of (Leblond et al., 2018) MT is also considered one of the possible fields of application for SEARNN. But it is different to the method proposed here as they are using each cell of a RNN as a regressor directly without an additional expert.

4. Methods

The method described in this work, ADBLEU, can be mainly split into two parts. The first is the oracle (expert) using the extracted lattices files from the Moses SMT system as a main source of knowledge. This knowledge is then used in the second part, the training process, to give the student additional information. The student's interfaces with the expert through queries of roll-outs or continuations which can be provided interactively or parallelized in large quantities. Through a data aggregation learning method the knowledge is transferred from the expert to the student. This is done iteratively to a point where the student out-performs the expert.

4.1. SMT Oracle (Expert)

The main target of ADBLEU is to improve the performance of the student's NMT model measured in BLEU. During the training of the student, the SMT Oralce is used as a the expert to provide additional information and guide partial translated sentences from the student in the right direction to maximize BLEU. The generation of lattices with Minimum Bayes-Risk (MBR) decoding and the linear BLEU approximation follows the method of (Tromble et al., 2008). The base implementation of the oracle was provided by (Sokolov et al., 2013) using the finite-state transducer library OpenFst by (Allauzen et al., 2007). It was then modified to the needs of this work, mainly to be able to fix the student's prefixes in lattice generation, work on large-scale datasets and to meet the state-of-the-art performance required during training.

4.1.1. Lattice Generation and Partial Translation

A translation word lattice is a compact representation for large N-best lists of translation hypotheses and their likelihoods. The lattice is a weighted acyclic finite state automaton containing states and transitions between them. Each transition has a word and a weight assigned to it. Candidate paths in the lattice consist of sequential transitions beginning at the initial and ending at a final state. Factorization of weights along a candidate path produces a total cost according to the model used, for example the BLEU approximation described later in section 4.1.2.
To show how a translation word lattice with a fixed prefix is generated a toy example is given below. Additionally visualization is given in figure 4.1. The reference and source sentence are denoted as R and S respectively. The prefix coming from the student is defined as P.

- S: die katze sitzt auf dem baum .
- R: the cat sits on the tree .
- P: my cat stands on

$P \subset R$ will denote that for the sequence of transitions in P there is at least one candidate path also containing these transitions starting from the initial state in R. Equally $P \not\subset R$

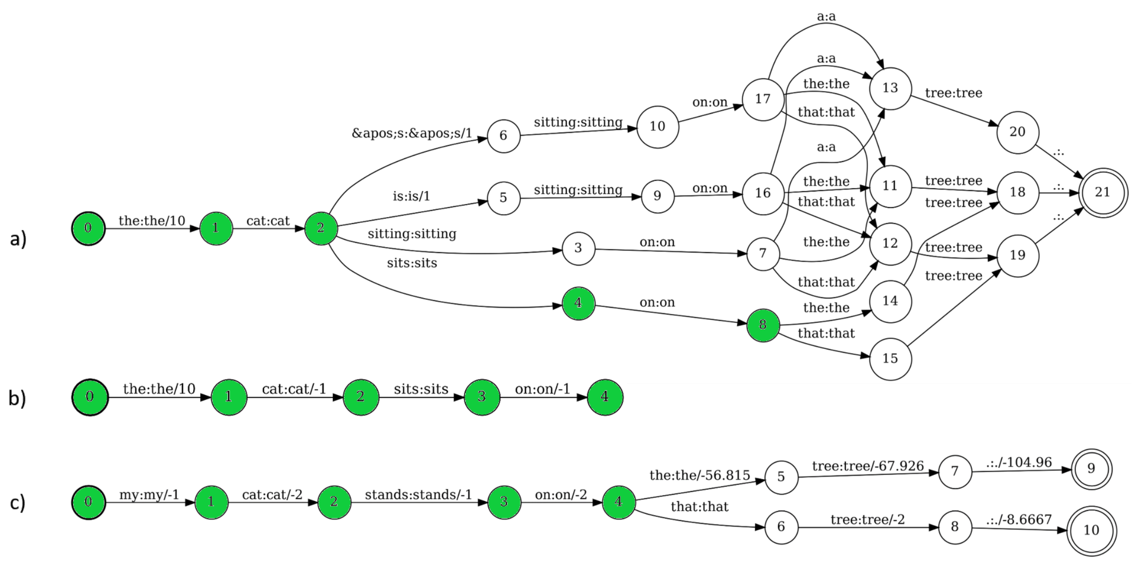


Figure 4.1.: Shows the process of finding the best translation word lattice given a fixed prefix. (a) Word lattice weight according to the prefix to find the closest representation of it in the lattice. (b) The best possible representation of the prefix. (c) Combination of the original word lattice and the prefix representation.

denotes that there is no path in R that contains P.
In this example $P \not\subset R$ was deliberately chosen to show how the oracle can react to prefixes coming from the student which are not in the translation lattice. To obtain a weighted word translation lattice the following steps are done:

1. Split the phrase lattice extracted by Moses into a word translation lattice **E_Moses** and assign the initial weight θ_0 to each transition.

2. Obtain **E_Partial** by re-weighting **E_Moses** based on the linBLEU approximation (section 4.1.2) with P as a reference, such that words that appear in P are being favored. **E_Partial** is shown in figure 4.1 (a), the sequence of green states shows the closest possible path to P according to the current lattice.

3. Extract **E_Partial_Path** from **E_Partial** as shown in figure 4.1 (b).

4. Re-weight **E_Moses** according to linBLEU again but this time with the original reference R and get **E_Final**.

5. Check if $P \subset$ **E_Final**:

   a) Yes: Go over all transitions in P and match them to the transitions in the according states in **E_Final** starting from the initial state. In the states that were passed delete all other transitions, leaving only the student's prefix as a starting sequence.

   b) No: Follow **E_Partial_Path** and delete the equal transitions in **E_Final** to obtain the point in the lattice where according to linBLEU is the closest end point to the prefix. Concatenate in front of this point a path of the student's prefix P as shown in figure 4.1 (c) and set it's first state as the only initial (starting) state.

6. Re-weight **E_Final** again according to the reference R to get the best possible translation word lattice given a fixed prefix P.

7. Find the n shortest paths where n is the number of unique paths from **E_Final**.

If $P \subset R$, P is forced as the only path for the prefix in the lattice. If $P \not\subset R$ the graph represents an adapted search space to a fixed prefix P, where the connection point of P and the lattice is the end of the best representation of P in the lattice. For example if a student starts translating a sentence and at some point does not know how to continue, the expert is asked how to continue from that point on. The expert reinterprets his idea of translating the sentence (shortest path in the original word lattice) and tries to find the best translation based on the student's prefix (shortest path in the prefix adapted lattice).

4.1.2. BLEU Approximation

The SMT oracle relies on the cooccurrences and the statistical correlation that Moses learned during training and ultimately output into the lattice files for each sentence. Each hypothesis build by the system contains a stack of phrases with their probabilities generated by the beam search. The objective of the SMT system is to maximize the probabilities (minimize the cost) for all sentences given the individual phrase probabilities as described in section 2.2.1.
As shown in the section 2.2.2 NMT systems are usually trained doing minimization on the cross-entropy loss function. It is continuous and differentiable, thus it can be used for back propagating gradients through the network and adjusting it's parameter.
Due to the counting of n-grams BLEU is not differentiable, so it can not be used directly when training a NMT model using gradient descent. This leads to the problem of expert and student being optimized in general for the same but mathematically for different objectives. Additionally the evaluation metric during testing which ultimately decides how good a model is, is yet another objective. Therefore providing additional information with the expert such that the student can learn from it, is non-trivial. To overcome the gap between the objectives a linear approximation of BLEU, which was originally introduced by (Tromble et al., 2008), is used to fine-tune the oracle. They show that the contribution of a single sentence can be approximated by computing a first order expansion of the change of log BLEU when removing that sentence from the corpus. The log BLEU implied by a single candidate **e** and a reference sentences **r** can be written as a linear function of the modified n-gram precision as follows:

$$\operatorname{lin BLEU}(\mathbf{e}) = \theta_0 \, |\mathbf{e}| + \sum_{n=1}^{N} \theta_n \sum_{u \in \Sigma^n} c_u\left(\mathbf{e}\right) \delta_u(\mathbf{r}) \tag{4.1}$$

Where $\theta_0, ..., \theta_N$ are parameters of the method, $c_u(e)$ is the number of times the n-gram u appears in **e**, and $\delta_u(\mathbf{r})$ is an indicator variable testing the presence of u in **r**: it is equal to 1 if the n-gram u appears in **r**, and to 0 otherwise (Sokolov et al., 2013). The second sum in the equation is equal to the modified n-gram precision p_n from the original BLEU score shown in section 2.3.2, equation 2.6 weighted by the parameter δ_n. The parameters

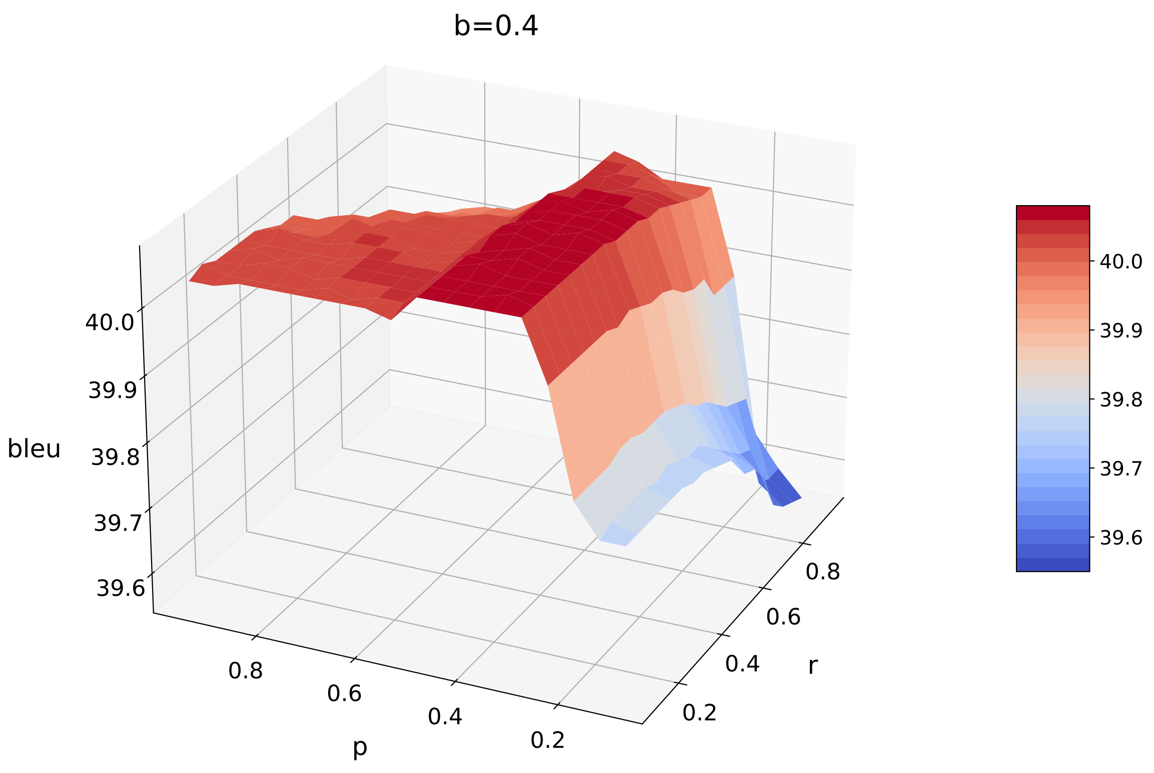


Figure 4.2.: Gridsearch for p and r with a beam threshold of $b = 0.4$ without teacher forcing.

can then be set to maximize the approximation of BLEU by using $N = 4$ following the method of (Tromble et al., 2008):

$$\theta_0 = 1, \theta_n = -\left(4p \cdot r^{n-1}\right)^{-1} \tag{4.2}$$

Where p and r are average values of the corpus unigram precision and common n-gram and (n-1)-gram ratio, respectively. The constant value of θ_0 roughly accounts for the brevity penalty and the values of θ_n are increasing rewards for matching n-grams (Sokolov et al., 2013).
To maximize linBLEU not only the best possible values for p and r have to be found but also different prefix lengths and beam thresholds need to be taken into consideration. The beam threshold as described in section 2.3.1 strongly affects lattice generation by pruning candidates translations. Additionally to simulate different translation states of the student the parameter need to be adjusted for various prefix length.
Equation 4.2 can not be solved for p and r analytically. Therefore a gridsearch was done for p and r from $0.1 - 0.95$ with a step size of 0.05 and beam thresholds ranging from $0.1 - 0.9$ with a step size of 0.1. Prefixes were taken from the reference with a length of $0\% - 80\%$ and a step size of 20%.

In figure 4.2 a gridsearch over all values of p and r for a beam threshold of $b = 0.4$ is shown. In this case no prefixes were used to show the maximum BLEU that can be achieved by the oracle with $b = 0.4$. The plot shows the BLEU approximations through

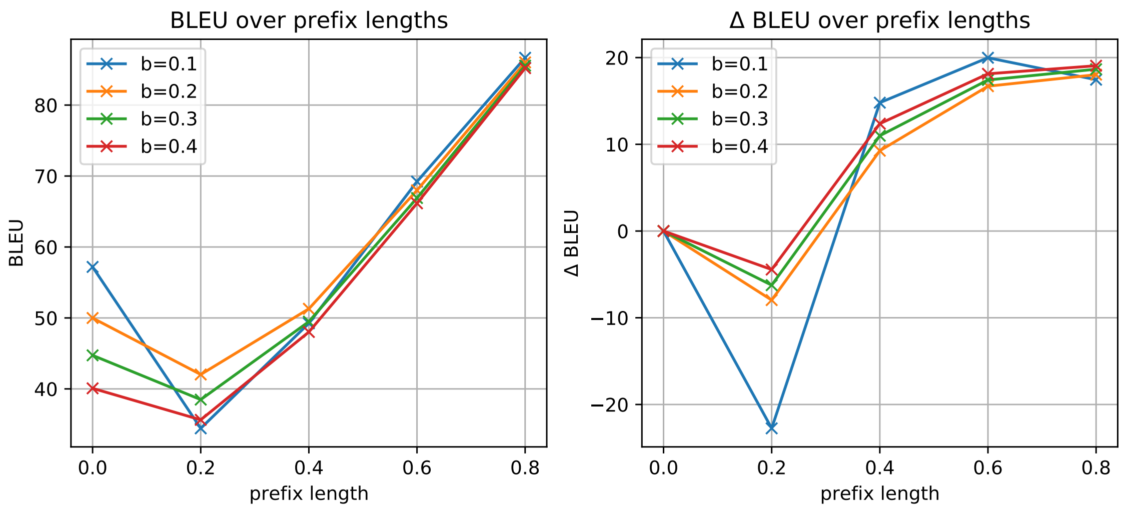


Figure 4.3.: Left: BLEU score over prefix lengths taken from the reference. Right: Difference of BLEU score from one discrete prefix length to the next. Each line represents a different amount of threshold pruning b for the beam search.

equation 4.2 depending on p and r. The colors are scaled to the minimum and maximum BLEU values of the data under consideration used for the plot.
From equation 4.2 it is trivial that $p > 0$ otherwise the term is 0. Independent of the beam threshold for every $p < 0.1$ the unigram precision contribution to the term is too low, such that the approximation fails to represent BLEU. r can be seen as a fine-tuning parameter for the n-gram precision ratio. To show how the best parameter for the approximation were extracted now the results for the gridsearches are reviewed.
Figure A.1 in the appendix shows the evaluations of four gridsearches of BLEU. The first and second row correspond to a beam threshold of $b = 0.1$ and $b = 0.2$ respectively. The first column shows the evaluation with no prefix and the second column with BLEU averaged over all prefixes sizes mentioned before. If for example the plot with $b = 0.1$ and no prefix is compared to the one in figure 4.2 the restriction that comes from the beam threshold becomes visible. It limits the flexibility during the beam search by pruning possible hypothesis in the process. Thus for increasing beam thresholds the BLEU values are squashed together more, eventually making p and r redundant. Furthermore figure A.1 shows in the second column that a higher beam threshold can correspond to better BLEU scores when using parts of the reference as prefix. This observation can be made when comparing $b = 0.1$ and $b = 0.2$ with the prefix length of 20% and 40%.

Figure 4.3 demonstrates the influence of the prefix length with regards to the beam thresholds on the BLEU. The left plot shows the BLEU score over all prefix length used. On the right the difference of BLEU from one discrete prefix length to the next can be observed. The only significant improvement of BLEU for $b = 0.1$ can be seen when using no prefixes, where it is almost 10 BLEU better than the gridsearch with $b = 0.2$. This is counter intuitive because every path that exists in the lattices of larger beam thresholds should also exists in the ones with less pruning. But as no MT system is perfect, Moses also introduces a domain gap between the target and the predicted distribution. When forcing the prefix into the lattice the flexibility of the larger lattices becomes a problem as the

beam search takes more hypothesis into consideration. The SMT system has more options to minimize the cost of the hypothesis but not necessarily maximize BLEU. Therefore it generates hypothesis that represent possible good translations according to Moses, but do match the reference suffix. With slightly less flexibility the lattice is limited, such that it is more likely to reconstruct the reference sentence. A trivial consequence for higher beam thresholds is that the reference suffix might not be in the lattice anymore. In this scenario it can be observed until 40% prefix lengths. Then the lattices become so inflexible from the relatively long prefixes that the beam threshold becomes less important.
After all the prefixes used in this evaluation are not "real" prefixes from the student. Using parts of the reference as a prefix is done for simulation and should only show how the beam threshold and different prefix lengths correlate with BLEU. To get a better understanding of how suboptimal prefixes influence the continuation performance, GoogleTranslate™ was used to generate good but imperfect translations w.r.t. the references. For an ideal interpretation multiple sets of prefixes would have to be gathered with different qualities of translations, to simulate the different translation qualities that the oracle encounters during the training of the student. This would result in an optimization problem where not only the gridsearch had to be done over the different prefix lengths for each p and r but also for prefixes with different translations quality. As an approximation for this an evaluation of the gridsearch for prefixes generated with GoogleTranslate will follow.
For this experiment the development set of IWSLT14 was used for evaluation. $N = 1953$ sentences were remaining after cleaning the corpora. The average sentence length is 20.38 words per sentence. The inferred sentences from the translator were cut uniformly random for $0-80\%$ of their length to obtain the prefixes $p \in P$. T defines the number of token in a sentence. The average value of T for the prefixes $\sum_{i=1}^{N} T(p_i)$ is 8.58 where the theoretical value is 8.15. This difference can be assumed as a normal deviation considering the size of the development set. It is assumed that the set of prefixes is a representative approximation for set of prefixes generated by a converged student.

In figure 4.4 the difference of BLEU for the oracle's continuations of the prefixes from GoogleTranslate and the reference are compared. Here the best possible beam threshold, in the setting, $b = 0.1$ was used. The drop in performance from reference to GoogleTranslate prefixes is almost 30 BLEU. This is because the oracle has to recover sentences that come from a completely different MT system with a larger vocabulary. Therefore the oracle encounters words never seen during training and the plots are not really representative for this case. The main reason the experiment was done is because of the shape of the curve, so how the parameter p and r change when working with imperfect prefixes. One can observe that the shape of the curves are almost equal. For the reference there is a slight decrease in BLEU from around $p = 0.5$ to $p = 0.3$ that is not visible in the scores of the other plot. The gridsearch was additionally done for the beam thresholds $b \in \{0.2, 0.3, 0.4\}$ to back up the findings here. They all show similar behavior and can be found in the appendix A.2, A.3 and A.4.

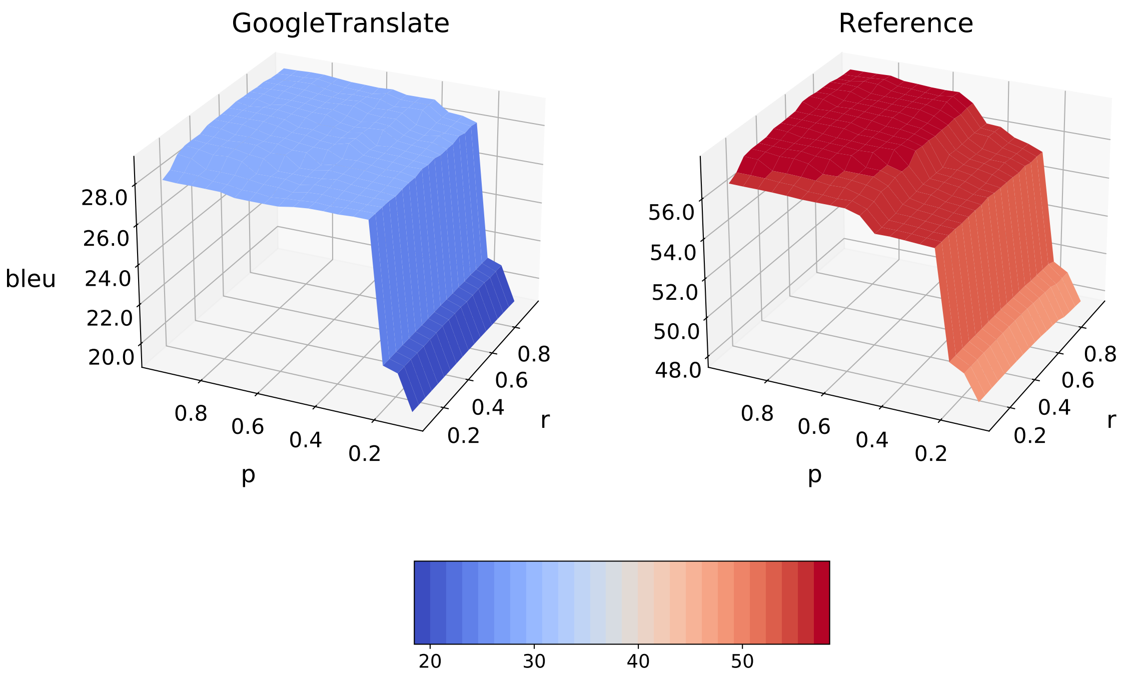


Figure 4.4.: Evaluation in BLEU of the oracle's recovery for imperfect prefixes with a beam threshold of $b = 0.1$.

4.1.3. Decoding Performance

Training time is a main metric when comparing MT systems. Large RNN based systems take weeks to train until convergence. The oracle is a crucial part of the training routine and is executed at least once for each instance in the training dataset. Therefore it's runtime needs to be evaluated carefully.
The oracle is loaded once for the whole training process. For each used beam threshold it creates shards of lattices, input and output symbols from the translations word lattices output by Moses. Shards limit the memory consumption at any given time during the decoding process, because not every lattice has to be loaded simultaneously. They are generated once for each dataset and beam threshold. The size and the number of sentences in total and in each shard is saved in a config file. When the oracle is loaded the first time and there is no existing config file the lattices are split into shards and the new config is saved. If the same dataset is loaded again the config file is validated and the already existing shards are used. The time it takes to create the shards will be denoted oracle's loading time.
When generating continuations with the oracle the most time consuming part is composing the lattices and finding the n shortest paths. When translating sentences they do not have dependencies between them, such that each one can be generated by an individual thread. Therefore this part is highly parallelized, so execution is fast. In figure 4.5 the memory consumption, loading and continuation time are shown over the beam threshold ranging from 0.1 to 0.9. Both y-axes used for memory in megabyte and time in seconds are drawn logarithmic to show the exponential effect of the beam threshold. It can be seen that the memory consumption decreases from around 1MB for a $b = 0.1$ to a 1/100 of it's original size (10KB) for $b = 0.9$. This also affects the oracle's loading and continuation time as

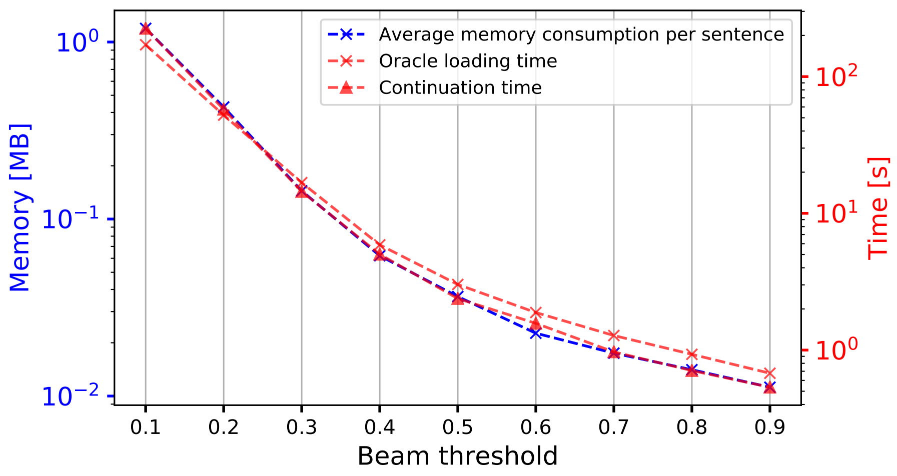


Figure 4.5.: Oracle's performance for memory consumption, loading and continuation time for each sentence.

the graphs consist of less states and transitions. When using the oracle as an expert in an IL environment both required BLEU performance and runtime should be taken into consideration.

4.2. NMT Model (Student)

Modeling sequence prediction tasks like MT is usually done with artificial neural networks by approximating a function that predicts a sequence of output tokens Y given an input sequence X. For a long time RNNs have been the architecture of choice when handling sequence data in MT tasks. An RNN has by design a recurrent part that needs to be computed in series. Training complex RNNs with millions of parameters on large scale datasets can take multiple weeks on state-of-the-art hardware. The urge for faster training times has led to extended research in alternative architectures. In 2017 the Transformer architecture by (Vaswani et al., 2017) was proposed. It completely replaces the convolutional and recurrent building block with self-attention as already stated in section 2.2.2. Additionally it reached new state-of-the-art performance while being more parallelizable. Therefore it was chosen as a neural network architecture that is representing the student in this work.

4.2.1. Design Choices

As a framework for training, testing and developing the student, Fairseq by (Ott et al., 2019) is used. It is a sequence modeling toolkit written in PyTorch that is fast and extensible. Additionally it provides support for mixed precision and multi-GPU training. The abstract implementation of Fairseq makes adding new modules easy. It was chosen over other NMT toolkits because it already includes most of the functionalities that were needed to build the framework of the ADBLEU method namely: optimizers, learning rate schedulers, predefined models, scoring and various search algorithms for decoding.
For ADBLEU shown in section 4.3 a new training routine, criterion and sequence decoder were implemented.

The interaction with the expert written in C++ is done via the pybind11 Python wrapper module by (Jakob et al., 2017). Pybind11 provides an easy to use interface for fast, paralellized C++ code. Exporting an object from C++ to Python for example, can be done by defining a wrapper object and instantiating it in Python. The wrapped C++ object can then be used like a normal object within Python. Pybind11 will interface almost all data structures like arrays, vectors and maps, such that the usage of C++ functions in Python is straightforward.

4.2.2. Baseline

The baseline is represented by a neural network using the Transformer architecture. It will act as a reference when converged and as a starting point for fine-tuning using AD-BLEU. The parameters of the Transformer were set according to the Fairseq model for the IWSLT14 translation task. The model is slightly smaller than the base Transformer model proposed by (Vaswani et al., 2017). That is, the encoder and decoder feed forward embedding dimensionality and number of attention heads reduces to half of its original size, thus from 2048 to 1024 and from 8 to 4 respectively.
The baseline is trained using inverse square root scheduling and the adaptive optimizer Adam (Kingma & Ba, 2017). For the scheduling a fixed amount of warmup updates are done. They work like normal updates during training but with a very small learning rate. It allows adaptive optimizers to estimate the way the gradient will take and therefore help pushing the network in the right direction. Inverse Square Root is a learning rate schedule $1/\sqrt{\max(n, k)}$ where n is the current training iteration and k is the number of warmup steps. This sets a constant learning rate for the first steps, then exponentially decays the learning rate until pre-training is over. Training is usually limited by an early stopping metric or a maximum number of updates. Patience is an example for early stopping that stops training after no improvement on the validation set was done for N consecutive epochs, where N is usually between $5 - 20$ depending on the dataset and the architecture of the network.
In section 5 a review of multiple baselines trained to various stages will be given to see how the expert in the IL algorithm can handle different qualities of baseline outputs. Furthermore a converged baseline will be used as a reference.

4.2.3. Decoding and Exploration Actions

During the inference of standard autoregressive NMT models new sentences are generated based on a token-level. To generate a so called hypothesis each output token is predicted using the input sequence and the previous predicted token. For each token of the output the input tokens are encoded into a context of embeddings. To decode a single token the attention mechanism scales these contexts based on the importance for the respective output tokens. Due to the complexity searching for the best token over the whole vocabulary with an exhaustive search is usually infeasible (the vocabulary is the same as the set of actions A and a token is equal to an action a). Thus using a beam search the n-best candidate tokens with the highest scores are considered. Where n can be set as a parameter, to either keep more or less candidate tokens in the selection. This way each hypothesis is progressively build and evaluated. At the end a list of finalized hypotheses is returned which can then be further evaluated through e.g. BLEU. The beam search is not described in more detail here as it was already illustrated in section 2.3.1 in the scope of SMT.

For ADBLEU one step deviations at a randomly sampled time t are done in the MDP. These so called exploration actions a_t are used as a point where the difference of expected cost-to-go $Q - \hat{Q}$ from the student and the expert is taken, where Q with the learned policy π is defined as $Q = \sum_t^T E_{s \sim d_\pi^t}[C_\pi(s)]$, equally the expert's expected cost-to-go can be defined with policy π^*. The training method and the loss function 4.3 of the method will be elaborated in section 4.3. The focus here will be the decoding process, where it is assumed that it is needed to decode a sentence up to the exploration action at time t. Therefore the sequential decoding/inference process is done to obtain the student's prediction $\hat{a}_{1:t}$ of the source sentence up till time step t of the output. For each output token in the sequence the network predicts the positional likelihood of each token in the vocabulary A. Taking an exploration action can be done using the following ways:

1. Take the most probable token according to the student (largest logit/odd). *This helps the student to exploit the oracle, useful for decreasing training time in the beginning.*
2. Take an uniformly random sampled token from A. This is what is done by (Ross & Bagnell, 2014) in AGGREVATE. *This results in slow training times but high generalization of the model.*
3. Create a stochastic exploration action, where a random token is sampled with a probability P and the most probable token is taken with $1 - P$. *This will decrease training speed in the beginning, but ultimately results in a more generalized model*
4. Use the oracle's representation for the sequence already predicted by the student $\hat{a}_{1:t}$. Continue this sequence with the oracle to obtain $a^*_{t:T}$. Force the student to take action a^*_t from the oracle. *By this the student is forced into the direction of the oracle. In later stages of the training when the student is almost as good as the oracle this might be necessary to still obtain useful information from the oracle.*

In principle there are many more ways to obtain an exploration action. The desired methods usually combines exploitation of the oracle for faster learning and exploration of the MDP for robustness of the model. For example this could be a scheduled alternative of case 3. where P dependents on the learning rate, such that in later stages of the training P is increased and the main source of knowledge during training goes from exploitation to exploration.
In this work cases 3. and 4. will be reviewed later, in section 5.

4.2.4. Data Aggregation

To receive expert's trajectories in the training process of the student an interface between them is needed. In general the student should be able to query the expert interactively, for

single instances and for whole datasets. As shown in section 2.4.2 imitation learning can be used to supply the learner with additional data. For the method proposed in section 4.3 the expert is used to create roll-outs $a_{t:T}$ given the student's prefix $\hat{a}_{1:t-1}$ for a randomly sample time t and any exploration action a_t. Given this method there are multiple options to generate different amounts of training data with the expert. In a sense of data aggregation the connection between the student and the expert can be seen as a $N \times M$ relation, where N and M are, respectively, the amounts of training instances that the student and the expert can create from a single prefix of the student $\hat{a}_{1:t-1}$. For example with a single prefix, $N = |A|$ training instances could be created by taking every possible exploration action $a_t \in A$ in the vocabulary. Then for each tuple of prefix and exploration action $(\hat{a}_{1:t-1}, a_t)$, $M = |C|$ roll-outs can be generated, where C is the set of unique paths in the word translation lattice of the expert. Additionally every time step in T could be used for exploration. Therefore there are $N \times M \times T$ possible data aggregations for a single instance (sentence pair) in the training set.
Furthermore datasets of examples collected in each epoch can be aggregated, like in DAGGER. Dataset D in epoch i would result in $D_i = D_i \bigcup D_{i-1}$, iteratively increasing the set of examples. Different procedures of data aggregation and their effect on the ADBLEU training method are evaluated in the results (section 5).

4.3. ADBLEU Training Method

In this section the main training method: ***A**ggregate **D**ata using approximated **BLEU** explorations* (ADBLEU) will be presented. First an overview will show the general procedure, the interaction between student and expert and the way the data is aggregated. Then the proposed algorithm will be analyzed in more detail. Two criterions, BLEU squared error (BSE) and Scaled BLEU Squared Error (S-BSE), will be introduced that should transfer knowledge from the expert's to the student's policy through BLEU differences. Additionally comparisons to some fundamental algorithms from section 2.4.3 will be made.

In figure 4.6 the training routine for ADBLEU is shown. For each epoch at first the original dataset is read by the student. The current policy creates the roll-ins by inferring all sentences and taking an exploration action at a randomly sampled time step. This aggregated data is then passed to the expert to continue the trajectories and obtain the roll-outs/training instances. The completed sentences are used for training the student. When training on the current dataset is done, it is omitted and the next epochs start with a new dataset.

The inverse reinforcement learning method proposed here uses data aggregation to recover an unknown reward function. It does so by generating reward-sensitive examples of states s, actions a, time t and the estimated future reward $\hat{Q}$ provided by the expert. For each input x in the dataset X one or multiple examples of $(s, a, t, \hat{Q})$ can be created in one epoch. Data generation is done by doing exploration actions once for each input, following the approach of AGGREVATE by (Ross & Bagnell, 2014). In contrast to AGGREVATE here a reward rather than a cost is used for future estimates.

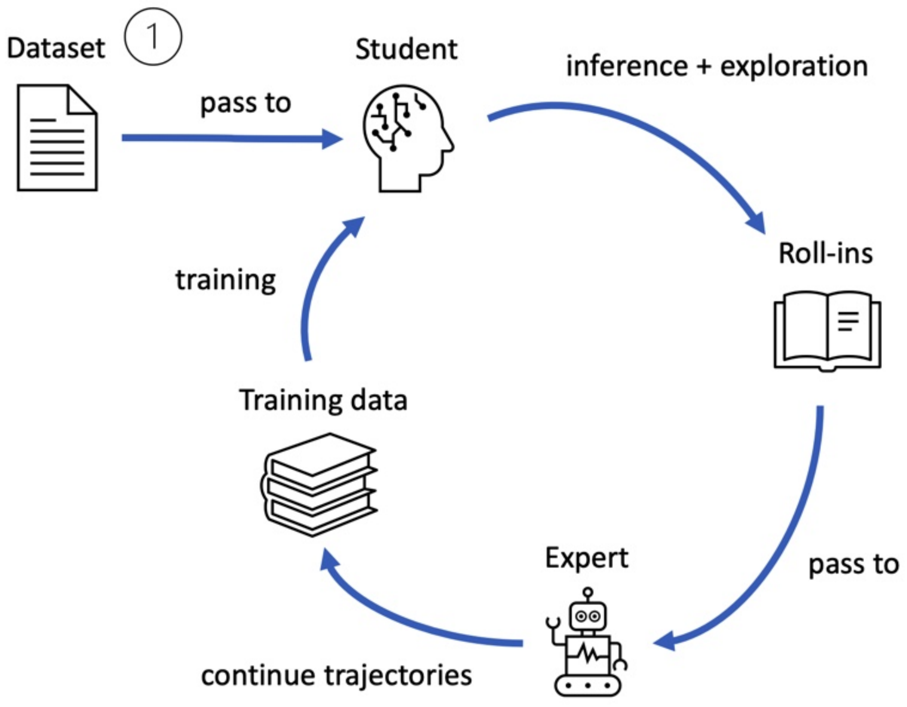


Figure 4.6.: Training routine with data aggregation using an expert. Every epoch starts with the original dataset at (1).

Algorithm 2 ADBLEU training routine with imitation learning and reward-to-go

```
Input: training dataset $X$, expert's policy $\pi^*$, exploration randomness $\beta$
Output: trained policy $\hat{\pi}_n$
 1: Initialize policy $\hat{\pi}_0$
 2: for i = 1 to N do
 3:   Dataset of examples $D = \emptyset$
 4:   for x in X do
 5:     Predict $\hat{\pi}_{i-1}(x) = \hat{a}_{1:T}$
 6:     Sample uniformly $t \in \{1, 2, ..., T\}$
 7:     if Probability $\beta$ then
 8:       Random exploration action $a_t = a \in A$
 9:     else
10:       Best exploration action $a_t = \hat{\pi}_{i-1}(s)$
11:     end if
12:     Predict $a'_{t+1:T} = \pi^*(x|\hat{a}_{1:t-1}, a_t)$ and observe reward-to-go $\hat{Q}$ at time $t$
13:     Add normalized reward-to-go difference $k(\sigma(Q(\hat{a}_{1:t})) - (\hat{Q} - Q(\hat{a}_{1:T})))^2$ to $D$
14:   end for
15:   Train policy $\hat{\pi}_i = argmin_{a \in A} \sum_j D_j(s, t, a)$
16: end for
```

Algorithm 2 shows the training algorithm of ADBLEU with imitation learning defined as a regression problem using the S-BSE loss. For simplicity a short form of k is defined here as $k \equiv k(\hat{Q}, Q(\hat{a}_{1:T}))$. The algorithm can be described as follows:
For each instance of the training dataset X the student's (learned) policy $\hat{\pi}_{i-1}$ of epoch i infers the sentence x (line 5). Then t is uniformly sampled from 1 to T where T is the length of the sequence. The inferred sentence $\hat{a}_{1:T}$ is partially used as a roll-in up to the point $t-1$. With probability β an exploration action a_t at time step t is taken completely random from all possible actions A. With probability $1-\beta$ the best possible action a_t at time step t according to the student is taken (line 7-11). The expert predicts the continuation $a'_{t+1:T}$ based on the partial roll-in $\hat{a}_{1:t-1}$ and the exploration action a_t. An example is obtained by taking the normalized squared difference of the student's and the expert's estimated future reward $k(\sigma(Q(\hat{a}_{1:t})) - (\hat{Q} - Q(\hat{a}_{1:T})))^2$. Where k is an indicator function that is 1 if $\hat{Q} \geq Q(\hat{a}_{1:T})$ and 0 otherwise. σ is a normalized scaling function that squashes the values in the range $(-1, 1)$ (e.g. tanh). $Q(\hat{a}_{1:t})$ is the student's reward-to-go starting at t. It is the maximum value of the unormalized output before the softmax layer (logit) of the neural network, that is the student. $\hat{Q} - Q(\hat{a}_{1:T})$ is the expert's reward-to-go starting at t normalized by the student's reward-to-go for the whole sequence.
This means that if the expert's reward-to-go estimate is lower than the student's ones, the whole equation is 0, thus the student does not learn from this instance because it is already better than the expert in this case. Additionally both terms are normalized either by the scaling function σ or by the differences of future reward estimates. σ is parameterized for the expected minimum and maximum values of $Q(\hat{a}_{1:t})$, such that the largest expected values are roughly mapped to 1 and the smallest to -1 so no information is lost. The first term $Q(\hat{a}_{1:t})$ estimates how good the student is with regard to the roll-in and exploration action. The second term $\hat{Q} - Q(\hat{a}_{1:T})$ estimates how good the roll-out of the student w.r.t. the expert is. In the end a squared difference is obtained that estimates how good the student's exploration action a_t was w.r.t the expert. The criterion is named S-BSE and is defined as in line (13):

$$\ell = k(\sigma(Q(\hat{a}_{1:t})) - (\hat{Q} - Q(\hat{a}_{1:T})))^2 \tag{4.3}$$

In contrast the proposed argmax regression criterion by (Ross & Bagnell, 2014) uses the unnormalized $Q(\hat{a}_{1:t}) - \hat{Q}$ and therefore does not account for future rewards of the expert being worse than the student's. The original criterion from their paper adapted for rewards rather than cost will be denoted as BSE. This rather complex objective is needed to directly optimize for the global objective - BLEU. As mentioned in the previous chapters the MLE training loss of the student is not always a good surrogate for the test error. Even if the test error is lowered, it is not a guarantee that the BLEU score is improved. BLEU in this instance can be seen as the hidden reward that is not directly known by the expert nor by the student. Additionally there is the discrepancy between the objective of the student (NMT) and the expert (SMT) that was reviewed before. By finding approximations of BLEU for the terms used in the loss and normalizing them, the method proposed here tries to directly optimize for the global objective.

Algorithm	Roll-in	Roll-out	#Roll-outs
DAGGER	π	π^*	$X \times T$
SEARN	π	π	$X \times T$
SMILe	π	π^*	$X \times T$
AGGREVATE	π	π^*	X
LOLS	$\hat{\pi}$	π	$X \times T$
ADBLEU	$\hat{\pi}$	π^*	X

Table 4.1.: Comparison of imitation learning algorithms with respect to their roll-in and roll-out policies.

To achieve this, at first the expert was optimized for BLEU using the approximation in section 4.1.2. Then both main terms of equation 4.3 were scaled into the range $(-1, 1)$ to make them comparable. Therefore for each of the student's interpretation of instance $x \in X$ a score is taken that measures the performance of the exploration action w.r.t. the expert and BLEU. This signal is given to the student, such that it learns what action is good and which is bad. The assumption can be made that in theory after N epochs the student learns to predict each action a_t of each instance $x \in X$ better or equally good as the expert.
Following **Theorem 2.2** from (Ross & Bagnell, 2014) this can be shown after N iterations collecting m regression examples with a probability of at least $1 - \delta$:

$$J(\hat{\pi}) \leq J(\pi^*) + 2\sqrt{|A|}T\sqrt{\hat{\epsilon}_{\text{class}} + \hat{\epsilon}_{\text{regret}} + O(\sqrt{\log(1/\delta)/Nm})} + O\left(\frac{Q_{\max}T\log T}{\alpha N}\right) \quad (4.4)$$

Where the total cost of executing π for T steps is denoted as $J(\pi) = \sum_{t=1}^{T} E_{s\sim d_\pi^t}[C(s, \pi(s))]$. $\hat{\epsilon}_{\text{regret}}$ denotes the empirical average online learning regret on the training examples. $\hat{\epsilon}_{\text{class}}$ denotes the empirical regression regret of the best regressor on the aggregated dataset. In theory this guarantee holds, in practice however using a sub-optimal oracle and BLEU differences there are many small deviations that make the algorithm perform differently. Results and evaluation for a real task will follow in section 5.

Now the question of why AGGREVATE was chosen as a basis algorithm in this method will be elaborated. Some of the algorithms that were reviewed in section 2.4.3 will be taken into consideration here. Mostly 3 factors need to be taken into account: Roll-in, Roll-out policy and number of times the expert gets queried in the method. In table 4.1 the algorithms are set side by side. Where π^*, $\hat{\pi}$ and π are the expert's, learned and mixture policies respectively. The stochastic policy π is usually defined as $\pi = \beta\pi^* + (1 - \beta)\,\hat{\pi}$ with learning rate β, such that the influence of the expert's policy decreases over time. The size of the dataset is denoted as X and the average sequence length of the instances in X is denoted as T. The algorithms DAGGER, SEARN, LOLS, and SMILe execute the expert for each action in each instance of X, thus doing $X \times T$ roll-outs. When considering the performance of the oracle proposed in section 4.1 this would result in a very slow training routine. Like AGGREVATE, ADBLEU only queries the expert ones for every sequence in X. Furthermore the learned rather than a stochastic policy is used for roll-ins as proposed by (Chang et al., 2015) (LOLS). They show that doing roll-ins with expert's policies is not preferable because it puts the student in a state that is already very good therefore might not be in the student's scope.

ADBLEU uses the learned policy as roll-in and the expert's policy as roll-out once for every sample in the dataset X. When thinking about training time this might only be reasonable when using an already decent student. In the case of MT where there is additional supervised training data available the student should be first trained on this using the MLE objective. For instance if the student is in the beginning of the training it's understanding of how to generate a sentence is weak. Then learning from a single step deviation, that is the exploration action, only gives the student a signal for the difference at that position of the sentence. The MLE objective on the other hand considers the loss of the whole prediction w.r.t. the reference. The amount of information is greater, thus the student learns faster. After the student is able to output good predictions ADBLEU can be used to improve the models capability to generalize and increase overall performance. The empiric results on this will be reviewed in section 5.

5. Results

This chapter will give an overview of the achieved results, the used data and the experiment process. Furthermore the results for the proposed training method ADBLEU with different variations of hyperparamters and data aggregation methods will be presented.

5.1. Data Preparation and Hyperparameters

The experiments were done on the IWSLT14 dataset for German to English translation. It is a dataset of a collection of human translated talks from the TED conference. TED talks are multidisciplinary in many different research topics and therefore not bound to a single domain of text. The dataset consists of the following sets with their respective number of sentences for each set in brackets: training ($160k$), validation ($7k$), testing ($5k$) and development ($2k$). It was then tokenized using the Moses tokenizer (Koehn et al., 2007) to separate punctuation from words using spaces and to replace symbols in words like *aren't* with *aren &apos;t*. After that it was lowercased and cleaned by omitting miss-aligned sentences and sentences with a length of more than 50 tokens in both source and target. A Byte Pair Encoding (BPE) vocabulary was created with subword-nmt (Sennrich et al., 2016) by iteratively counting symbol pairs and replacing the most frequent with a new symbol. This merged symbols represent character-level n-grams and form a new vocabulary. The joint source-target vocabulary size was set to 32000 unique BPE tokens. The dataset was then encoded using the generated BPE vocabulary. This design choice was made because of the rather small dataset size of IWSLT14.

The fundamental hyperparameters for training the Transformer model with MLE and fine-tuning it with ADBLEU are mostly the same, as shown in table 5.1. Equal parameters during fine-tuning are adopted from the baseline and denoted with a #.
For training the baseline 4000 warmup updates with the inverse square root scheduling were used. Warmup updates work like normal updates during training but with a very small learning rate. It allows adaptive optimizers to estimate the direction the gradient will take and therefore help pushing the network in the right way. Inverse Square Root is a learning rate schedule $1/\sqrt{\max(n, k)}$ where n is the current training iteration and

Parameter	Baseline	Fine-tune
Learning Method	Supervised	Inverse Reinforcement
Criterion	Cross Entropy	Scaled BLEU Squared Error
Optimizer	Adam ($\beta_1 = 0.9$, $\beta_2 = 0.98$)	#
Learning Rate	5×10^{-4}	5×10^{-7}
Learning Rate Scheduler	Inverse Square Root	Fixed
Dropout	0.3	#
Weight Decay	1×10^{-4}	#
Warmup Updates	4000	0
Token per Batch	4096	#

Table 5.1.: Fundamental hyperparameters for training a baseline model with MLE and fine-tuning with ADBLEU.

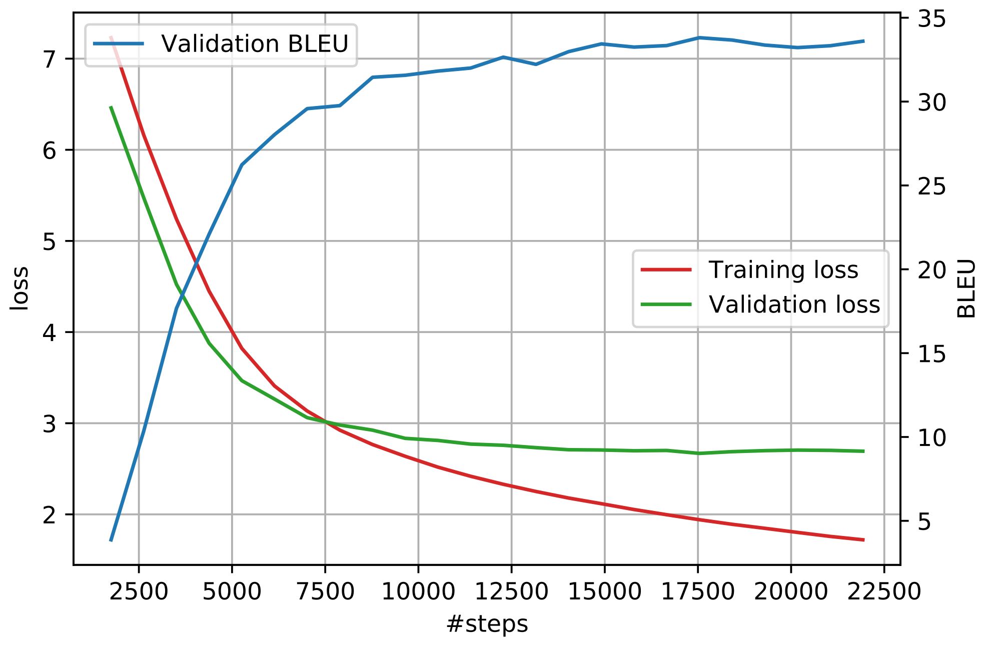


Figure 5.1.: BLEU performance, training and validation loss over the course of training the baseline represented by a Transformer model. The hyperparameters are set according to table 5.1.

k is the number of warmup steps. This sets a constant learning rate for the first steps, then exponentially decays the learning rate until pre-training is over. For fine-tuning, the baselines hyperparameters are reloaded from it's last state. The learning rate schedule is changed to a fixed schedule together with a smaller learning rate of 5×10^{-7} to do only minor adjustments to the weights when using the BSE or S-BSE criterion.

5.2. NMT Baseline

In this section the results for the experiments made to answer the main research question will be presented. The performance will be validated in different tasks, metrics and convergences states of the baseline and the after fine-tuning with ADBLEU.

In figure 5.1 the performance of the baseline Transformer during training is shown. The model converged after 25 epochs with a patience of 5 or about $22k$ steps with 4096 tokens processed in each step. The plot shows the loss as the sum of the negative log-likelihood on training and validation set. After 20 epochs or $17.5k$ steps the best performance, in terms of the lowest loss on the validation set, is achieved. In the following sections different states of the baseline e.g. after $5k$, $10k$ or $17.5k$ steps (i.e. best performance during convergence) will be used as a starting point for fine-tuning with ADBLEU or as a reference to compare to. In the figure the discrepancy between the training and global objective becomes visible. As the training and validation loss are still decreasing to the end of the plot, the BLEU score stops increasing at some point. Further elaboration on this will be done in the discussion (chapter 6).

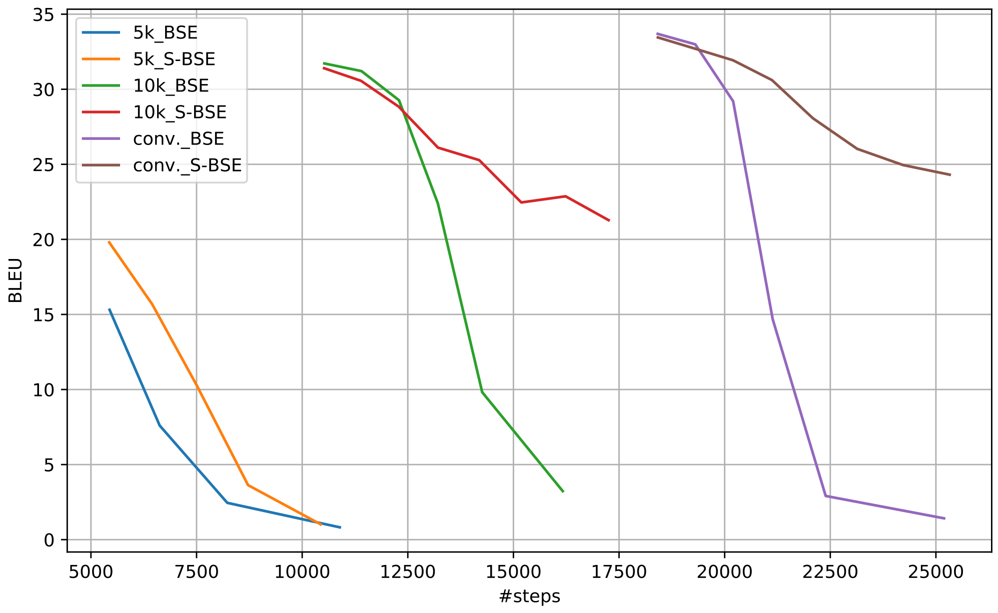


Figure 5.2.: BLEU performance on validation set during fine-tuning using the ADBLEU method with different starting points and IRL objectives.

5.3. Inverse Reinforcement Learning BLEU Objectives

Here the training results using the inverse reinforcement learning objectives for direct BLEU learning BSE and S-BSE will be shown. Both can be used interchangeably as a criterion in the training process of ADBLEU. As elaborated in section 4.3, BSE corresponds to the regression transformation explained in the paper of (Ross & Bagnell, 2014) using BLEU directly. Furthermore S-BSE is a scaled version of this criterion which is normalized between $(-1, 1)$ and only takes data points into account which the student can learn from, i.e. where the expert's expected reward-to-go is higher.

Figure 5.2 shows the effect of fine-tuning the baseline model from the starting points $5k$, $10k$ and $17.5k$ steps respectively. The experiments have been made for both proposed IRL objectives and were stopped after 8 hours of fine-tuning. Both objectives are decreasing the BLEU score on the validation set with increasing number of training iterations. For all of them a trend was visible and therefore no additional steps were needed. To further analyze this trend of BLEU decreasing with increasing number of steps, another experiment was done. This time using an even smaller learning rate of 10^{-7}, the S-BSE objective and fine-tuning for $15k$ steps after $17.5k$ steps of baseline training. Then a comparison of the training and validation loss was done to show the overarching problem of the method. This is illustrated in figure 5.3. It becomes visible that the training objective (S-BSE) correlates with the validation objective (MLE) at first but after around $7.5k$ steps they diverge.

To understand the problem of the method further investigation was done by logging additional parameters. For this the most representative and logical scenario, that is, having a converged baseline and fine-tuning it with the S-BSE objective, was used as an example. The fine-tuning process of *conv._S-BSE* will be analyzed in more detail. It was first intro-

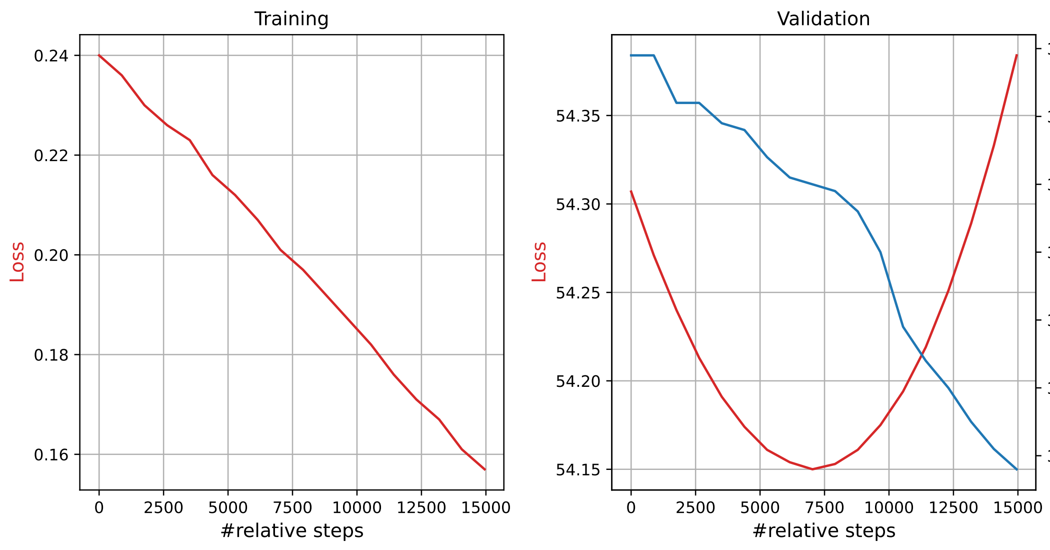


Figure 5.3.: Training, validation loss and validation BLEU of the ADBLEU method using the S-BSE objective for $15k$ steps of fine-tuning after baseline convergence.

duced in figure 5.2 for the illustration of the change of BLEU over the number of steps. The following parameters were additionally logged over the course of the training for the whole training set and each individual batch:

- B_o: The average BLEU score of the oracle.
- B_s: The average BLEU score of the student.
- B_{oe}: The average BLEU score of the oracle's using the exploration action of the student.
- The ratio of B_o/B_s which shows for how many sentences the oracle's continuations were better than the student's inferred sentences. This corresponds to the indicator function of the S-BSE objective in equation 4.3.

In figure 5.4 the BLEU scores of the oracle's, student's and the oracle using student's exploration actions are displayed. Additionally B_o/B_s shows for how many of the sentences in the training set the oracle's continuation was better (in terms of BLEU) than the student's prediction. As the student's performance gets worse by training on sub-optimal examples, B_o/B_s increases. Furthermore the correlation between the score of the model taking the exploration action (exploration logit) and the BLEU score of the sentence taken and averaged over the training set is shown. This was measured using the Pearson correlation and an exploration randomness (probability of taking a completely random action from the vocabulary) of 0.1 and resulted in a value of $0.04 - 0.07$. This means there is basically no correlation between the exploration action taken and the BLEU score of the sentences that are used as references during training. Further evaluation on the effect of this will be discussed in chapter 6.
As translations cannot only be measured by the precision of n-gram overlap additional metrics were used to measure the performance. In table 5.2 the results for the evaluation on the IWSLT14 test set with four additional metrics are shown. The two models that are

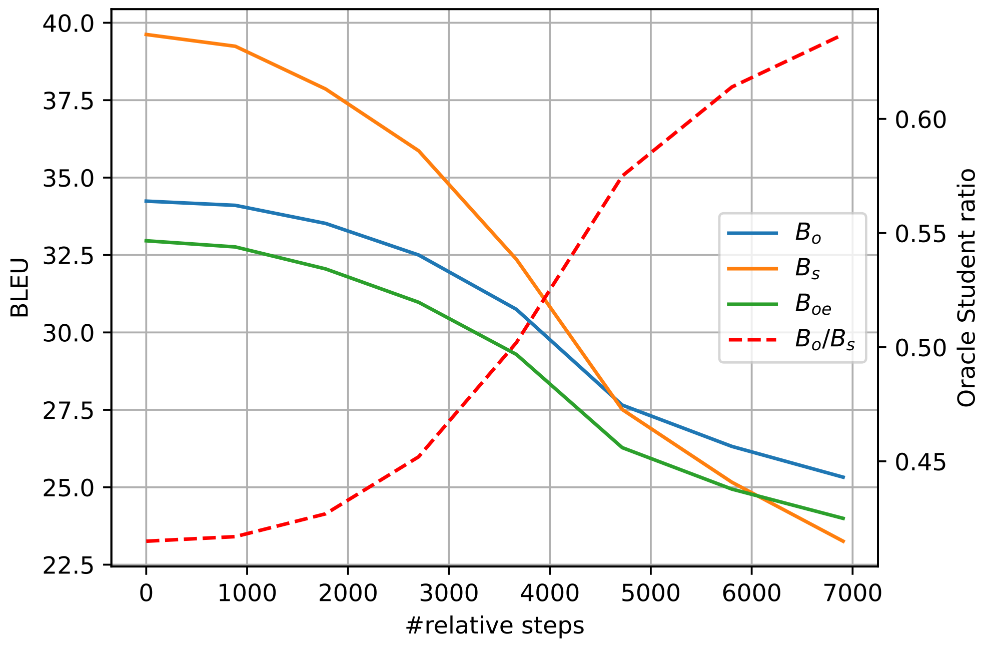


Figure 5.4.: BLEU performance of student(B_s), oracle(B_o), oracle with student exploration action(B_{oe}) and the ratio of oracle to student BLEU(B_o/B_s) over the course of fine-tuning the model.

Metric	Baseline (17.5k)	Fine-tuned best (26.3k)	Fine-tuned last (34.2k)
BLEU	**34.34**	34.06	33.66
GLEU	**0.665**	0.663	0.66
METEOR	**0.57**	0.569	0.564
BLEURT	0.17	**0.173**	0.172
Perplexity	6.11	**6.06**	6.11

Table 5.2.: Comparison of the baseline and a fine-tuned model trained with ADBLEU using the S-BSE objective on the test set. The best model is evaluated according to the NLL loss on the validation set.

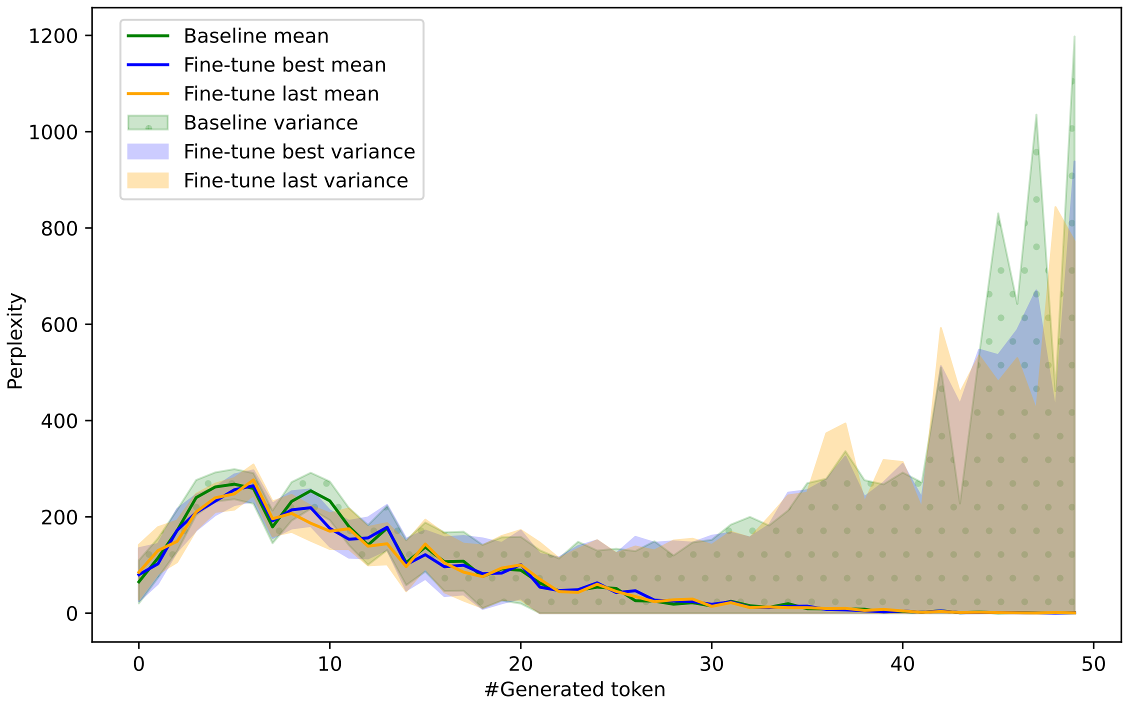


Figure 5.5.: Mean and variance of perplexity over number of generated tokens for baseline and fine-tuned models.

compared against the baseline are fine-tuned using ADBLEU and the S-BSE criterion. The *Fine-tuned last* model is fine-tuned for additional $34.2k - 17.5k = 16.7k$ steps. Where the *Fine-tuned best* was trained for $26.3k - 17.5k = 8.8k$ steps and selected by the minimum NLL loss on the validation set from the models during fine-tuning.
GLEU, METEOR and BLEURT were introduced in section 2.3.3. They are metrics that are closer to human judgement of machine translated text. The perplexity shows how certain the model is at predicting the respective tokens. Based on the definition done in section 2.2.1 it can be calculated using the cross entropy between the predicted and the reference distribution as an exponent for the basis 2. Therefore a lower perplexity corresponds to the model being more certain about what token to take and is therefore better.
In the experiments the best fine-tuned model exceeds the performance of the last fine-tuned model in all of the measured metrics. Compared to the baseline the best fine-tuned model performs slightly better in BLEURT and perplexity but fails to improve BLEU, GLEU and METEOR. Based on these findings another experiment displayed in figure 5.5 was conducted. It shows the mean and variance of the perplexity for each consecutive token for the test setting explained above. The plot shows that the models are more uncertain in the beginning of generating a sequence and get more certain after around 10 tokens. This is expected as for the initial tokens the self-attention mechanism has less tokens from the output to condition on. The high variance after 30 generated tokens can be explained by the few samples actually reaching that length. Additionally auto-regressive models get increasingly uncertain with more generated tokens. The consequences thereof will be further explained in the section 6.1.

Oracle Beam Thresh.	s_BLEU	s_GLEU	BLEU	GLEU
0.1	14.88	12.69	30.69	32.82
0.2	14.10	12.13	29.43	31.88
0.3	12.73	11.06	27.95	30.75
0.4	11.71	10.25	26.88	29.88

Table 5.3.: Quality of oracle's actions grouped by beam threshold and measured in forms of BLEU and GLEU. The prefix s_ denotes the suffix version of the corresponding metric.

5.4. Data Aggregation

The way data is aggregated directly influences the learning process of the student, as mentioned in section 4.2.4. Therefore evaluations for several hyperparameters that influence data aggregation are shown here. For instance, creating large aggregated datasets each epoch results in a less flexible system, whereas smaller datasets let the student learn more dynamic, due to the knowledge that is inferred every epoch.
ADBLEUs data aggregation can be adjusted using hyperparameters. The influence of these hyperparameters on the fine-tuning process was approved on the validation set of IWSLT14. Following hyperparameters were taken into consideration for generating data:

- **Beam Threshold:** This limits the number of hypotheses based on a relative factor of the probability of the best hypothesis (reviewed in detail in section 4.1.2).
- **Number of Continuations:** The count of continuations generated for each sentence. The BLEU approximation sometimes fails to represent the real BLEU score. Multiple continuations can help to find the best sentence according to BLEU by considering multiple unique paths in the word translation lattice.
- **N-gram Order:** The number of n-grams that are taken into consideration when approximating BLEU with the oracle.
- **Exploration Randomness:** The probability by which a random token from the vocabulary is used as the exploration action.
- **N-best Explorations:** The n-best tokens at the exploration position according to the student.

In figure 5.6 a correlation matrix of the data aggregation parameters and the scoring metrics is visualized, where -1 or blue is strongly anti-correlated and 1 or orange is strongly correlated. Based on the findings in the correlation matrix, the parameters that are most influential are analyzed further in the tables 5.3 and 5.4.

Where the quality of oracle's and student's exploration actions measured in BLEU, GLEU and their respective suffix metric is shown. A suffix version of a metric is calculated by taking the difference of the score of the hypothesis and the prefix with regard to the reference. Formally:

$$s\_g(h, p, r) = g(h, r) - g(p, r) \tag{5.1}$$

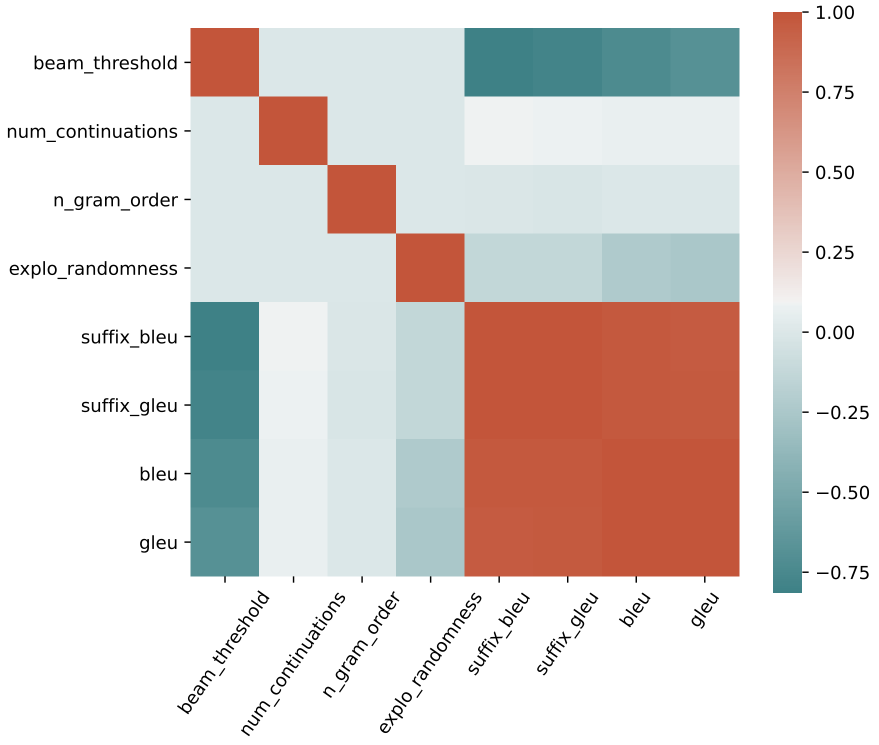


Figure 5.6.: Correlation of data aggregation hyperparameters with scoring metrics.

Student Beam Thresh.	Explo. Rand.	s_BLEU	s_GLEU	BLEU	GLEU
0.1	0.1	13.59	11.56	29.16	31.58
	0.3	13.42	11.41	28.28	30.72
	0.5	12.86	10.97	27.19	29.74
0.2	0.1	13.01	11.17	28.23	30.87
	0.3	12.44	10.70	27.27	29.98
	0.5	11.89	10.26	26.12	28.95
0.3	0.1	11.66	10.10	26.86	29.84
	0.3	11.38	9.89	26.05	29.06
	0.5	10.79	9.42	24.83	27.96
0.4	0.1	10.76	9.38	25.95	29.07
	0.3	10.32	9.06	24.90	28.13
	0.5	9.88	8.70	23.88	27.20

Table 5.4.: Quality of student's exploration actions grouped by beam threshold, exploration randomness and measured in forms of BLEU and GLEU. The prefix s_ denotes the suffix version of the corresponding metric.

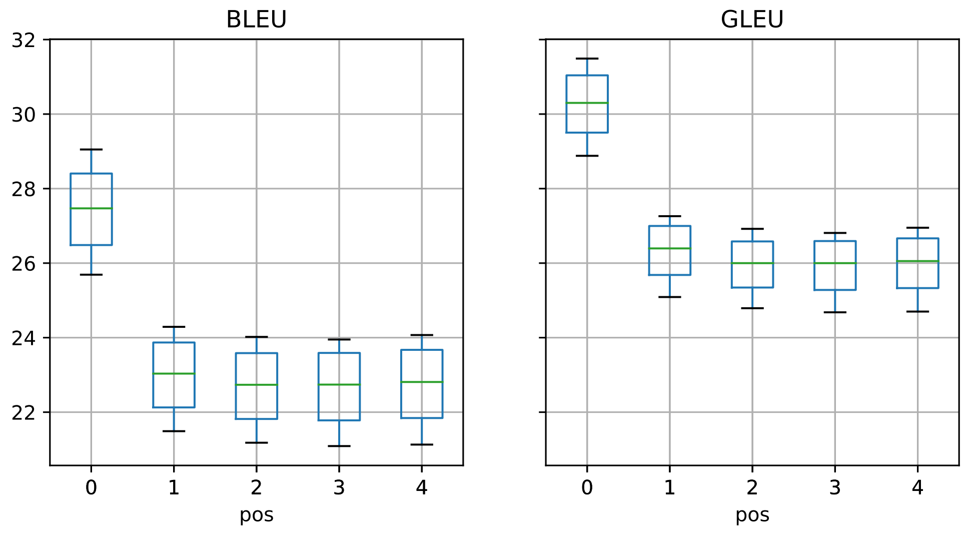


Figure 5.7.: Variation of BLEU and GLEU score for the 5-best exploration actions of the student. Pos corresponds to the respective position in the descending sorted n-best list.

Where the function $g()$ is the scoring metric, h is the hypothesis under consideration, r is the reference and p is the prefix of the hypothesis. The resulting function $s\_g()$ is an approximation of the suffix by the corresponding scoring metric with regards to the reference. In other words, it gives an indication how good the suffix generated, by the oracle, is with respect to the reference.
The exploration randomness corresponds to the probability P_t of taking a completely random action/token from the vocabulary at time step t as the exploration action. Table 5.4 shows that the scores are barely influenced by the exploration randomness.

In figure 5.7 the statistical distribution of the 5-best exploration actions by the student in BLEU and GLEU is given. The box-plot shows the distribution of the data in quartiles for each position in the 5-best list. A box with it's whiskers is divided into quartiles which are bound by 5 values with define 4 areas in the plot. The first range Q_0 is set by the minimum and visualized by the lower whisker. Q_1 represents the lower edge of the box and is defined by the median of the lower half of the dataset. The line in the middle of the plot is the median or Q_2. Similar to Q_1, Q_3 is displayed as the top edge of the box and is set by the median of the upper half of the dataset. Finally Q_4 is the top whisker and shows the maximum. Additionally outliers can be shown by crosses outside of the $Q_0 - Q_4$ range.

6. Discussion

This chapter will cover the detailed evaluation of the achieved results, the chosen methodology and the validity of the experiments. First the results from chapter 5 will be analyzed and the baseline is matched against the fine-tuned model in different configurations. Finally in section 6.2 the chosen objectives, algorithms and the overall experiment setup will be evaluated and compared to other inverse reinforcement learning approaches.

6.1. Results interpretation

In this section the evaluation of the obtained results for the baseline and fine-tuned model, using the proposed data aggregation methods, is done. At first investigation will be done on the data aggregation method of ADBLEU which can be adjusted using different parameters. These parameters are essential for the performance of the expert and therefore also crucial for the overall training process. In the method the student learns directly from the differences of the expected BLEU reward-to-go during training. If the difference of expert's and student's expectation is high the student learns faster. To achieve this the expert has to create decent continuations for the student's prefixes at any given position in the sentence. This is why so much optimization went into the hyperparameters of the SMT expert and the data aggregation method.
In figure 5.6 the correlation of some of these hyperparameters is shown. As expected the beam threshold pruning parameter is almost directly anti-correlated to the scoring metrics. From the evaluation of section 4.1.3 it is known that there are computational limitations for decreasing this parameter. To create continuations in feasible time for all sentences in the training set the beam threshold must be set to 0.2 or higher. Surprisingly the n-gram order, which defines the n-grams that the precision for the word translation lattices is calculated on, is also anti-correlated. For this measurement only the n-gram order of three and four were taken into consideration which means that the n-gram order of three is beneficial for the overall performance. The number of continuations on the other hand has almost no effect on the score, whereas the exploration randomness shows again an anti-correlated behavior, which is expected due to the fact that random tokens used in exploration generally perform worse in terms of BLEU. Based on this finding further research was done on the most relevant parameters, the beam threshold and the exploration randomness. The results are shown for oracle's/expert's exploration in table 5.3 and for student's exploration in 5.4. The student's exploration is done by concatenating the student's prefix $\hat{a}_{1:t-1}$ with the most probable token based on the student's expectation at time step t and the oracle's continuations $a_{t+1:T}$. Now considering the probability P_t at which a random token would be taken as an exploration action. For instance, the average BLEU score on the validation set using a probability $P_t = 0.1$ is 30.87. When changing only this parameter to $P_t = 0.5$ the resulting BLEU score changes to 28.95. If now the way how BLEU is computed is considered, it is clear that P_t only changes the probability of this very token to contribute to the n-gram precision, but not to way the oracle interprets the sentence. This difference in BLEU really shows how inflexible the oracle is to alternating

prefixes and exploration actions. Despite the student generating it's most probable or a random token, the oracle will generate the same continuation most of the times.
For the oracle's exploration the student's prefix $\hat{a}_{1:t-1}$ is concatenated with the most probable token based on the oracle's expectation at time step t and the oracle's continuations $a_{t+1:T}$. In this case doing random explorations does not make any sense because the vocabulary is the same for the oracle and the student, therefore a random exploration from the student and the oracle would result in the same outcome and was already done for the student.
Furthermore a validation on the n-best exploration tokens from the student was done to display the error that is incurred when not using the best exploration token. Due to the way BLEU works with the concept of counting n-grams, it is expected that everything that is deviating from the reference results in a drop of the score. But when considering that the GLEU score also behaves in the same way and is supposed to correspond to fluency rather than n-gram overlap, the finding done previously is additionally underlined. The oracle is inflexible and will generate the same continuations no matter what exploration token it gets from the student. This problem will strongly influences the overall results and will be shown in the next paragraph.

In figure 5.1 the most important metrics over the course of training are shown for the baseline. After the first epoch ends at around $2k$ steps, the initial data point is recorded. The peak BLEU 34.5 and lowest validation loss 2.66 is taken at $17.5k$ steps. This was the starting point for most further fine-tuning using the ADBLEU method. In non-convex optimization it can often be the case that a local optimum in the loss function is interpreted as a global optimum. If the optimization is stuck in a local optimum, it can be hard to find a way out. It was tried to systematically excluded this problem by using multiple starting points in terms of number of steps that the baseline was trained for, such that in principle a global optimum could have been found. Figure 5.2 shows six measurements that were done to obtain the main results showing the global objective, BLEU, that shall be maximized. The starting points $5k$, $10k$, and $17.5k$ steps of baseline training, were used respectively to exclude that the optimization is stuck in a local rather than a global optimum. Additionally starting from more than one baseline shows how the training performance develops for different qualities of outputs of the student. Three measurements were done for both objectives BSE and S-BSE using the same hyperparameters. The initial data point for all curves begins after the first epoch of training with the respective objective. First the results of the BSE objective will be evaluated.
It is noticeable that for all starting points BLEU heavily decreases in a rather small amount of steps. Starting from $10k$ and $17.5k$ shows the same behavior of decreasing BLEU slowly for two epochs and then going down rapidly. It is somewhat different for the starting point of $5k$ steps where it instantly declines at a high rate. This difference can be explained by the changing qualities of prefixes that the student produces and which are then forwarded to the expert to complete them. Additionally the used learning rate is possibly set to high, which would lead to a strong change of the parameters of the model and therefore would explain the rapid decline of BLEU. As the BSE objective was only used in the early phases of the project no more hyperparameter tuning was done for it. Therefore it

might not have been optimized to the best possible extend. It was eventually completely replaced by the scaled version, S-BSE, as it yielded better results in all tests done. It is still worth mentioning as the BSE objective directly corresponds to the argmax regression definition in (Ross & Bagnell, 2014) for their cost-to-go objective, thus showing that it is not applicable to this context of directly optimizing for BLEU, is still a relevant finding.
Similar to BSE, S-BSE completely fails to train the student when the prefixes are of bad quality. This is acceptable, as by definition the data aggregation method of ADBLEU is intended to be used for fine-tuning an already decent system due to the exploration and the single step deviations. Doing exploration with a system that is still very perplex of what token to take in each position, will lead to slow training in general. For the starting points of $10k$ and $17.5k$ the fine-tuning process shows a smoother behavior which can be broken down to a near optimal learning rate and qualitatively good prefixes generated by the student. Still the model fails to improve the BLEU score on the validation set.
Based on this further investigation was done using the converged baseline at $17.5k$ steps and fine-tuning it with the S-BSE objective and an even smaller learning rate of 10^{-7} for now $15k$ steps. This should approximate how the method behaves in the limit using tiny updates. The results for this measurement are shown in figure 5.3. It shows that for the first $7.5k$ training steps the training and validation loss are both decreasing. Surprisingly this does not affect the BLEU score on the validation set, showing once more the discrepancy between the negative log-likelihood or cross entropy loss and BLEU.
Over the course of this work BLEU showed it's benefits and drawbacks multiple times. On the one hand it is a common metric for measuring the quality of MT output and is fast to calculate. On the other hand it is not differentiable and the correlation to human judgement in comparison to other metrics is quite low, as mentioned in section 5.3. Therefore additional metrics where used to verify the quality of the natural generated language by the baseline and the fine-tuned model using ADBLEU. Table 5.2 shows that even if BLEU decreased more complex metrics which are closer to human judgement can increase over the baseline. Considering the rather small difference between the baseline and the fine-tuned model these improvements could have probably been achieved with training the baseline for the same amount of steps on the regular dataset without data aggregation. This additional experiment has not been done and is left for future work.

6.2. Method reflection

The data aggregation and inverse reinforcement learning method proposed in this work - ADBLEU - tries to tackle the problem of machine translation and it's discrepancy of the surrogate negative log-likelihood loss during training and the global objective of BLEU during testing. Various examinations were made to find the optimal parameters for the method. Starting with a SMT oracle that uses the approximation of BLEU from word translations lattices with Minimum Bayes-Risk decoding in section 4.1.2. The free parameters of the function that approximates BLEU were found by doing a gridsearch on the IWSLT14 corpus that was later used to train the student. Additionally the decoding parameters of the SMT system were evaluated carefully to find a compromise between

decoding speed and hypothesis quality. The oracle was tuned, such that it outputs the best possible translation with regards to the BLEU approximation in feasible decoding time. Measurements, like the gridsearch in figure 4.2, showed that the SMT system is able to output good translation hypothesis with regards to BLEU, in general. In comparison to the baseline the oracle is able to output sentences that have a higher BLEU score, even when the word translation lattice is pruned to obtain better decoding speed. This was first shown in figure 4.3, were it can be observed that the oracle is able to continue sentences starting with a reference prefix well, resulting in outputs that scored 40 BLEU or higher for all of the beam thresholds ranging from 0.1 to 0.4. This performance can only be achieved when doing **full** translations or when the reference is given as a prefix. When **continuing** a prefix of a sentence generated by a third-party system, like the student, the oracle mostly fails to recover the sentence and output quality degrades massively. Further investigation was done to simulate non-perfect prefixes by the use of the conventional online translation system GoogleTranslate. Figure 4.4 showed that even for the best feasible beam threshold of 0.1 the system is not able to come close to it's original performance.
In the context of using the oracle as an expert in an imitation learning scenario the assumption was made that additional data that coming from the expert can still be beneficial in the training process. As (Chang et al., 2015) has shown that with the right policy management during training the student can achieve above expert performance. Therefore the ADBLEU algorithm was introduced to generate additional data with the choice of two different inverse reinforcement learning objectives: BSE and S-BSE. Both try to directly optimize the model for the best possible BLEU score. With default parameters the ADBLEU algorithm follows, on a basic level, the one from (Ross & Bagnell, 2014) with a slight variation in roll-in policy. As shown by (Chang et al., 2015) instead of using a stochastic policy of expert and student it is beneficial to use just the student's policy for roll-ins. Expert roll-in policies put the student in an already too good state, such that it does not learn from it's own mistakes during training.
When aggregating training data many different methods and hyperparamerters can be used. All try to create additional examples based on the knowledge of the expert to train the student. For example the method of DAGGER continuously increases the size of the dataset by concatenating examples from previous epochs. Therefore the knowledge from the previous epochs is not only given to the current epoch by adjusting the parameters of the model but also through old examples that are still in the dataset. This makes the system inflexible and hard to train on real world datasets.
In AGGREVATE and ADBLEU for each sentence pair in the dataset a new data point is generated every epoch. Exploration is done through taking one step deviations at randomly sampled time step t within the sequence. Each aggregated dataset is only used for one epoch. When using an already pretrained system it therefore stays more flexible and potentially converges faster. On the other side using methods like ADBLEU or AGGREVATE to train a system from scratch result in long and costly training times. As already pointed out by (Cheng et al., 2018) to train a policy fast it is advisable to first learn on imitation, or if at hand supervised data, and then do further exploration with reinforcement learning. In the case of MT, there is plenty of supervised training data

available for most languages. Training a student on the supervised data and then continuing the training with imitation learning using the ADBLEU method with a SMT expert seems logical. In a real world scenario however the student can only learn from examples which actually provide something to learn from, i.e. having a higher BLEU score than the student's prediction itself. Through the use of surrogate losses and approximations an additional loss of information is infused. Therefore the student is only able to learn from examples which are a lot better than it's own prediction measured in BLEU. This problem is especially visible when using the unnormalized BSE objective, which uses every data point generated from student's prefix, exploration action and oracle's continuation no matter how good or bad it is. Figure 5.2 shows this problem very clearly: the better the performance of the student the faster it degrades due to bad continuation from the expert. Based on this a normalized and scaled version of the BLEU-Squared-Error was proposed. The S-BSE objective tries to overcome the previously mentioned problems by selecting training examples where the oracle's continuation is better then the student's. This decreased the rate at which the system degraded but the student still could not learn from the cherry picked examples.
Figure 5.4 essentially shows the problem of the whole process. Starting from a converged baseline the student is better on 75% of examples from the training data. Therefore only 25% of the oracle's continued sentences are used as new training data. As the student's performance degrades over time also the oracle's continuations get continuously worse, as they are partly based on it (by the prefix). At some point the oracle is performing better on 60% of training dataset but the student still can not learn from these examples because they are only marginally better. Now looking back at the problem, that the oracle is performing bad when the prefix differs to much from the original reference - and the circle closes.
The oracle is limited by the prefix performance of the student, in the best case the oracle outputs continuations that are on average slightly better then the ones from the student. But the student is only able to learn from examples that are a lot better due to the surrogate loss. This creates a process of continuously degrading the student's performance with regards to BLEU. In other more complex MT metrics the exploratory data aggregation can improve the performance of the student as shown in table 5.2.

After all ADBLEU is a data aggregation method that tries to bring the objective during training and testing time of MT models closer together. It does so by trying to iteratively improve the student's predictions by making it knowledgeable of it's own mistakes and giving adaptive references from a SMT expert. In general the method of inverse reinforcement learning through data aggregation has shown it's potential throughout this work. Using a more versatile expert that is less bound to the quality of the prefix would result in better continuations. With better training examples the student would most probably improve over the baseline. This thought will be continued in section 7.2.

7. Conclusion

Now an answer to the research questions will be given based on the evaluations and interpretations made in the last chapter. Additionally a final conclusion will be drawn and possible improvements and modifications will be shown for future work.

7.1. Conclusions

This research was aimed to solve the question of: Can a state-of-the-art neural machine translation student be improved by using a statistical machine translation expert in an imitation learning scenario?
For this experiment the IWSLT14 English to German corpus was used. The expert was build by using a SMT system that is based on word translation lattices which are weighted using a linear BLEU approximation. To evaluate the free parameters of the function a gridsearch was done to get the best approximation with regards to BLEU. The learning part or student of the experiment was represented by a neural network using the state-of-the-art Transformer architecture. Transferring knowledge from the SMT expert to the NMT student was done through the novel data aggregation algorithm ADBLEU that was introduced in this work. It is based on partial translation and exploration actions from the student and continuations from the expert. Through parameterization the method can be adjusted to different needs. In general the data is aggregated by first inferring each input sequence of the training data with the student to a randomly sampled point t. Then depending on the parameters of the method a single exploration action is taken at point t either randomly, based on the student's or the expert's belief what the best action might be with regards to the future expected reward. In the case of MT this action corresponds to a single token or word. Given the inference of the student, concatenated with the exploration action, the expert is queried to continue the trajectory. The difference of the student's and the expert's belief how to continued the sentence measured in BLEU is used as a loss function. In essence the knowledge is transferred to the student through the difference of BLEU of the expert's and the student's way of continuing a translation started by the student.
Two optimization objectives BSE and S-BSE were introduced to directly train the student on the global objective - BLEU. BSE represents the original argmax regression objective proposed by (Ross & Bagnell, 2014). It uses every aggregated data point to infer a loss, no matter how the quality of the continuation is. S-BSE is a modified and normalized version that scales the output and only takes data points where the expert is providing better continuations than the student. In several evaluations throughout this work S-BSE showed better performance than BSE and was therefore also used for the final results. Together with the flexible data aggregation concept of ADBLEU, the S-BSE objective can be seen as one of the main contributions of this work.
As proposed by previous work it is not advisable to train the student with a data aggregation method like this from scratch. The initial output of the student directly correlates with the amount of knowledge that is transferred, by the oracle, for each sample. For

instance, if the student generates every bad quality sentences the expert will not be able to provide useful information. Using a pretrained student and fine-tuning it with the additional data from the method to improve the overall performance is the main purpose of the method.
In different experiments throughout this work, considering many alternative data aggregation techniques, it was shown that the method, as it is, has a major problem: The SMT expert is not flexible enough to react to the student's prefixes and exploration actions. A single experiment was particularly revealing where the BLEU performance of the expert's continuations were measured with changing levels of randomness on the student's exploration action. It was shown that the probability at which a random token would be taken as an exploration action would hardly affect the resulting BLEU score of the continuation. In further investigation it was revealed that this is caused by the expert mostly generating the same continuation no matter what the prefix is. This inflexibility of the expert has shown to be the main problem of the method.
Additionally other experiments provided in this work have displayed that the SMT expert is not able to improve the performance of an already trained NMT student. Therefore in this setting the overarching research question can be answered by saying: It is not possible to improve a NMT student with a SMT expert due to fact that the expert is not adaptive enough to react to mistakes made by the student. This means that the student cannot learn from it's mistakes based on the expert's continuations. Consequently, the maximum quality difference between the student's and the expert's continuation is limited, so that very little additional information that can be generated by the expert.

7.2. Future work

This section will show how the findings from the last section can be used in future work to improve the proposed method. It will be also shown how to create a system with slight modifications that can use this concept for enhanced learning.
To improve the overall performance of the method a NMT rather than a SMT expert could be used. Using an architecture with more parameters than the baseline and training it on a larger corpus of text would result in a more flexible expert that could be used as a drop in replacement. This way a more generalizable expert would be provided to possibly overcome the issues mentioned in the last section. In contrast to the SMT expert it could actually help the student to recover from it's mistakes and not only suggest the same continuation independent of the student's output. Through the simple and generalized interface provided in the implementation done for this research, essentially any expert that generates natural language, conditioned on the student's output, could be used.
Additionally scheduled hyperparameters for the data aggregation method of ADBLEU could be introduced to improve training. For instance, setting the exploration randomness based on the learning progress of the student would simplify the training for the student in the beginning. This would bring the training closer to supervised learning in the beginning and later would be increasingly like reinforcement learning. As suggested by previous works this helps to improve training time when starting from scratch and increases the overall

performance later.
Furthermore, the progress made in recent years by for example (Cheng et al., 2018; Leblond et al., 2018; Sun, Venkatraman, et al., 2017) in the field of policy gradient methods could be used for imitation learning. These methods usually use RNNs because each cell of the RNN can be seen as a single regressor. When using the same architecture for student and expert the differences in each cell for the same input could be used for imitation learning, where for each token in the sequence the corresponding cell would output a multi-dimensional representation of the student's and expert's belief directly. Therefore the difference of the expert and the student can be used for training without using a surrogate loss. Then the policy instead of the value-function is optimized directly to overcome several known problems.
Additionally to further validate the proposed methods, the experiments could be conducted on larger datasets. It would be interesting to see if the data aggregation on larger datasets result in better generalization of the model, or worse. An open question is therefore if applying a larger dataset on the same problem could yield similar but different results.

Bibliography

Abbeel, P., & Ng, A. Y. (2004). Apprenticeship learning via inverse reinforcement learning, 1. https://doi.org/10.1145/1015330.1015430

Alammar, J. (2018). The illustrated transformer. Retrieved April 18, 2021, from http://jalammar.github.io/illustrated-transformer/

Allauzen, C., Riley, M., Schalkwyk, J., Skut, W., & Mohri, M. (2007). Openfst: A general and efficient weighted finite-state transducer library. *Proceedings of the 12th International Conference on Implementation and Application of Automata*, 11–23.

Bahdanau, D., Brakel, P., Xu, K., Goyal, A., Lowe, R., Pineau, J., Courville, A., & Bengio, Y. (2017). An actor-critic algorithm for sequence prediction.

Bahdanau, D., Cho, K., & Bengio, Y. (2016). Neural machine translation by jointly learning to align and translate.

Banerjee, S., & Lavie, A. (2005). METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, 65–72. https://www.aclweb.org/anthology/W05-0909

Bengio, S., Vinyals, O., Jaitly, N., & Shazeer, N. (2015). Scheduled sampling for sequence prediction with recurrent neural networks.

Bengio, Y., Ducharme, R., Vincent, P., & Janvin, C. (2003). A neural probabilistic language model. *J. Mach. Learn. Res.*, *3*(null), 1137–1155.

Bordag, S. (2008). A comparison of co-occurrence and similarity measures as simulations of context. *In 9th CICLing*, 52–63.

Chang, K.-W., Krishnamurthy, A., Agarwal, A., au2, H. D. I., & Langford, J. (2015). Learning to search better than your teacher.

Cheng, C.-A., Yan, X., Wagener, N., & Boots, B. (2018). Fast policy learning through imitation and reinforcement.

Daumé, H., Langford, J., & Marcu, D. (2009). Search-based structured prediction. *CoRR*, *abs/0907.0786*. http://arxiv.org/abs/0907.0786

Debao, C. (1993). Degree of approximation by superpositions of a sigmoidal function. *Approximation Theory and its Applications*, *9*(3), 17–28. https://doi.org/10.1007/BF02836480

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding.

Hayes, G., & Demiris, Y. (1995). A robot controller using learning by imitation. *Citeseer*, *676*.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, *4*(2), 251–257. https://doi.org/https://doi.org/10.1016/0893-6080(91)90009-T

Ipek, E., Mutlu, O., Martínez, J. F., & Caruana, R. (2008). Self-optimizing memory controllers: A reinforcement learning approach. *2008 International Symposium on Computer Architecture*, 39–50. https://doi.org/10.1109/ISCA.2008.21

Isaac, A., & Sammut, C. (2003). Goal-directed learning to fly., 258–265.

Jakob, W., Rhinelander, J., & Moldovan, D. (2017). Pybind11 – seamless operability between c++11 and python [https://github.com/pybind/pybind11].

Kalchbrenner, N., & Blunsom, P. (2013). Recurrent continuous translation models. *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 1700–1709. https://www.aclweb.org/anthology/D13-1176

Kingma, D. P., & Ba, J. (2017). Adam: A method for stochastic optimization.

Koehn, P. (2013). Moses Core Algorithm. Retrieved March 7, 2021, from http://www.statmt.org/moses/?n=Moses.Background

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., & Herbst, E. (2007). Moses: Open Source Toolkit for Statistical Machine Translation.

Leblond, R., Alayrac, J.-B., Osokin, A., & Lacoste-Julien, S. (2018). Searnn: Training rnns with global-local losses.

Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., & Khudanpur, S. (2010). Recurrent neural network based language model. *Proceedings of the 11th Annual Conference of the International Speech Communication Association, INTERSPEECH 2010*, *2*, 1045–1048.

Murphy, K. P. (2013). *Machine learning : a probabilistic perspective*. MIT Press. https://www.amazon.com/Machine-Learning-Probabilistic-Perspective-Computation/dp/0262018020/ref=sr_1_2?ie=UTF8&qid=1336857747&sr=8-2

Mutton, A., Dras, M., Wan, S., & Dale, R. (2007). GLEU: Automatic evaluation of sentence-level fluency. *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, 344–351. https://www.aclweb.org/anthology/P07-1044

Ng, A. Y., & Russell, S. J. (2000). Algorithms for inverse reinforcement learning. *Proceedings of the Seventeenth International Conference on Machine Learning*, 663–670.

Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Grangier, D., & Auli, M. (2019). Fairseq: A fast, extensible toolkit for sequence modeling.

Papineni, K., Roukos, S., Ward, T., & Zhu, W. J. (2002). Bleu: A method for automatic evaluation of machine translation. https://doi.org/10.3115/1073083.1073135

Pomerleau, D. A. (1989). Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems 1* (pp. 305–313). Morgan Kaufmann Publishers Inc.

Ranzato, M., Chopra, S., Auli, M., & Zaremba, W. (2016). Sequence level training with recurrent neural networks.

Ratliff, N. D., Bagnell, J. A., & Zinkevich, M. A. (2006). Maximum margin planning. *Proceedings of the 23rd International Conference on Machine Learning*, 729–736. https://doi.org/10.1145/1143844.1143936

Ross, S., & Bagnell, D. (2010). Efficient reductions for imitation learning. *Journal of Machine Learning Research - Proceedings Track*, *9*, 661–668.

Ross, S., & Bagnell, J. A. (2014). Reinforcement and Imitation Learning via Interactive No-Regret Learning. *CoRR*, *abs/1406.5*. http://arxiv.org/abs/1406.5979
Aggrevate

Ross, S., Munoz, D., Hebert, M., & Bagnell, J. (2011). Learning message-passing inference machines for structured prediction. *Proceedings / CVPR, IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2737–2744. https://doi.org/10.1109/CVPR.2011.5995724
Dagger

Sellam, T., Das, D., & Parikh, A. P. (2020). Bleurt: Learning robust metrics for text generation.

Sennrich, R., Haddow, B., & Birch, A. (2016). Neural machine translation of rare words with subword units.

Shannon, C. E. (1948). A Mathematical Theory of Communication. *Bell System Technical Journal*. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x

Shimosaka, M., Kaneko, T., & Nishi, K. (2014). Modeling risk anticipation and defensive driving on residential roads with inverse reinforcement learning, 1694–1700. https://doi.org/10.1109/ITSC.2014.6957937

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, *529*(7587), 484–489. https://doi.org/10.1038/nature16961

Sokolov, A., Wisniewski, G., & Yvon, F. (2013). Lattice bleu oracles in machine translation. *ACM Transactions on Speech and Language Processing (TSLP)*, *10*. https://doi.org/10.1145/2513147

Sun, W., Bagnell, J. A., & Boots, B. (2018). Truncated horizon policy search: Combining reinforcement learning & imitation learning.

Sun, W., Venkatraman, A., Gordon, G. J., Boots, B., & Bagnell, J. A. (2017). Deeply aggrevated: Differentiable imitation learning for sequential prediction.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (Second). The MIT Press. http://incompleteideas.net/book/the-book-2nd.html

Tesauro, G. (1992). Practical issues in temporal difference learning. *Machine Learning*, *8*(3–4), 257–277. http://www.cs.ualberta.ca/~sutton/tesauro-92.pdf

Translator, M. (n.d.). Machine Translation. Retrieved March 7, 2021, from https://www.microsoft.com/en-us/translator/business/machine-translation/#research

Tromble, R., Kumar, S., Och, F., & Macherey, W. (2008). Lattice {M}inimum {B}ayes-{R}isk Decoding for Statistical Machine Translation. *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, 620–629. https://www.aclweb.org/anthology/D08-1065

Turovsky, B. (2016). Found in translation: More accurate, fluent sentences in Google Translate. Retrieved March 7, 2021, from https://www.blog.google/products/translate/found-translation-more-accurate-fluent-sentences-google-translate/

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need.

Wu, L., Zhao, L., Qin, T., Lai, J., & Liu, T.-Y. (2017). Sequence prediction with unlabeled data by reward function learning. *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 3098–3104. https://doi.org/10.24963/ijcai.2017/432

Yandex Blog. (2017). One model is better than two. Yandex.Translate launches a hybrid machine translation system. Retrieved March 7, 2021, from https://yandex.com/company/blog/one-model-is-better-than-two-yu-yandex-translate-launches-a-hybrid-machine-translation-system/

Zens, R. (2008). *Phrase-based Statistical Machine Translation: Models, Search, Training* (Dissertation). RWTH Aachen University.

Zhang, J., Utiyama, M., Sumita, E., Neubig, G., & Nakamura, S. (2017). Improving neural machine translation through phrase-based forced decoding. *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 152–162. https://www.aclweb.org/anthology/I17-1016

Appendix

A. Additional Gridsearches

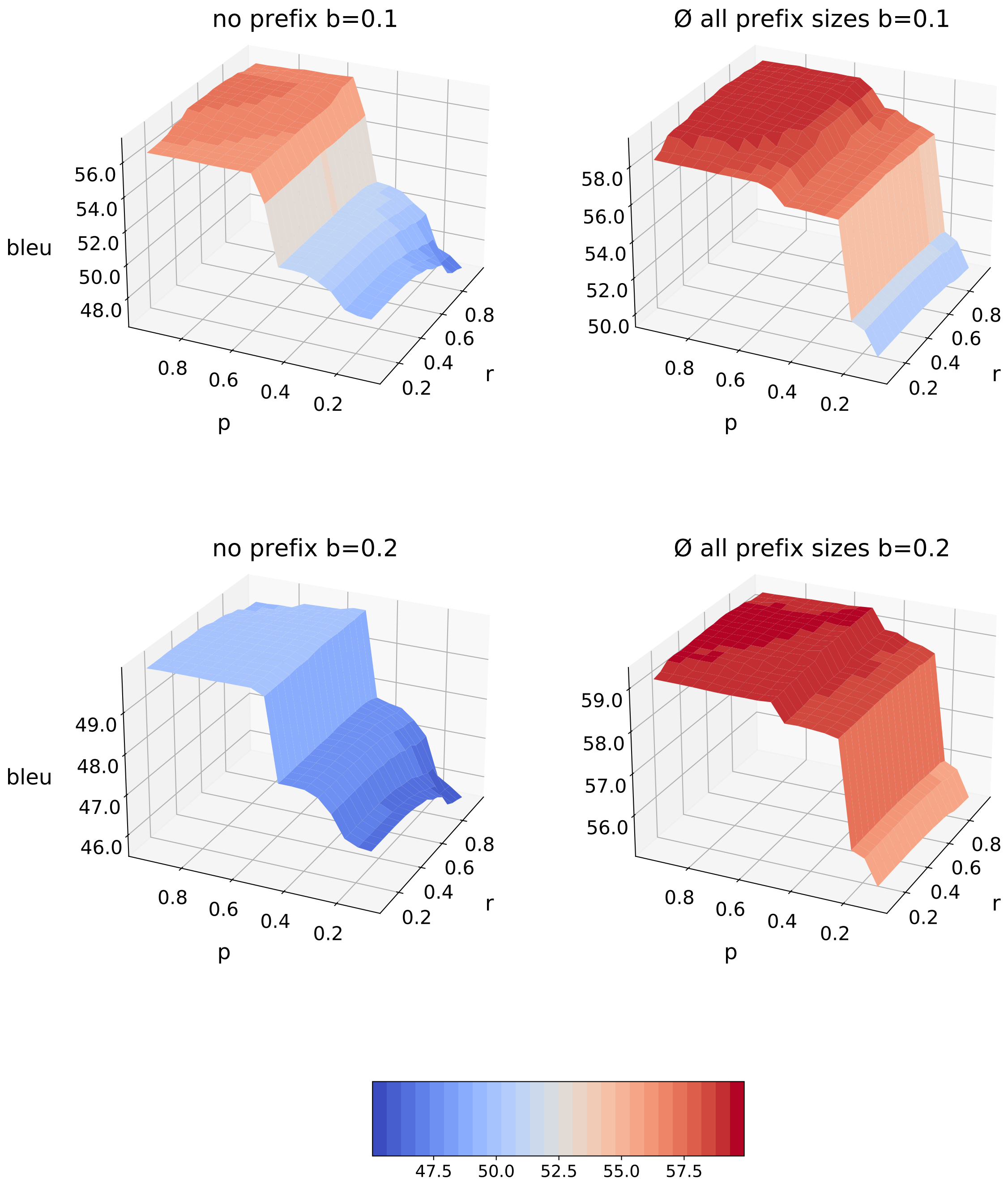


Figure A.1.: Comparison of gridsearches with respect to their beam threshold and prefix sizes, evaluated on BLEU over p and r.

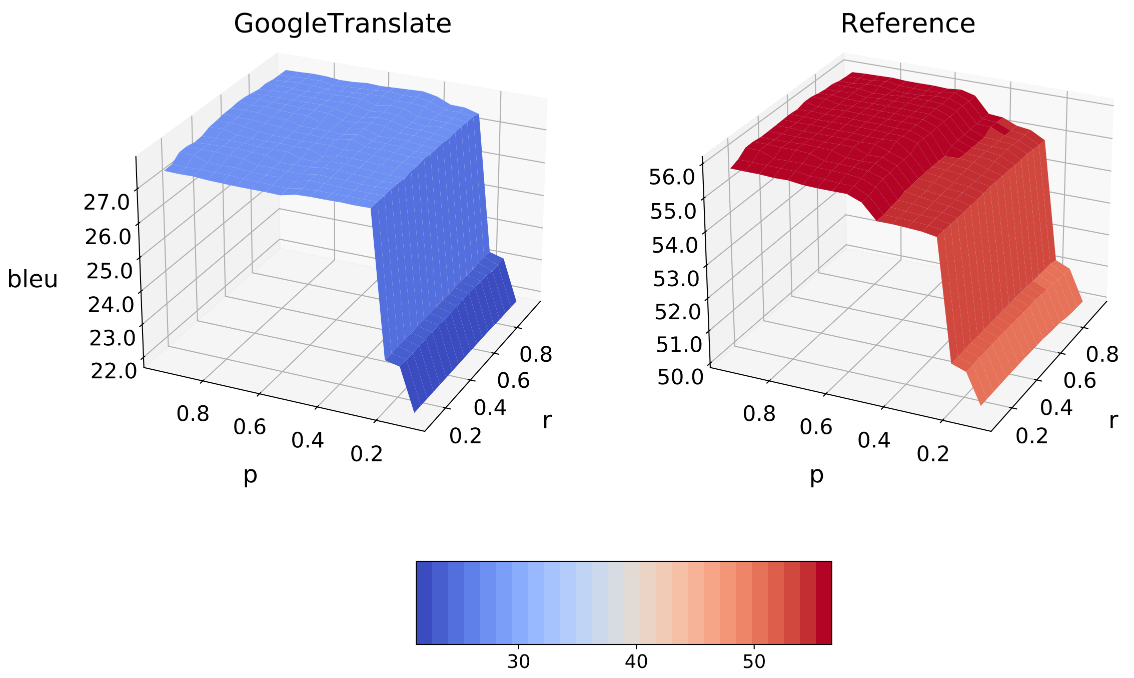


Figure A.2.: Evaluation of BLEU performance of oracle recovery for imperfect prefixes with a beam threshold of $b = 0.2$.

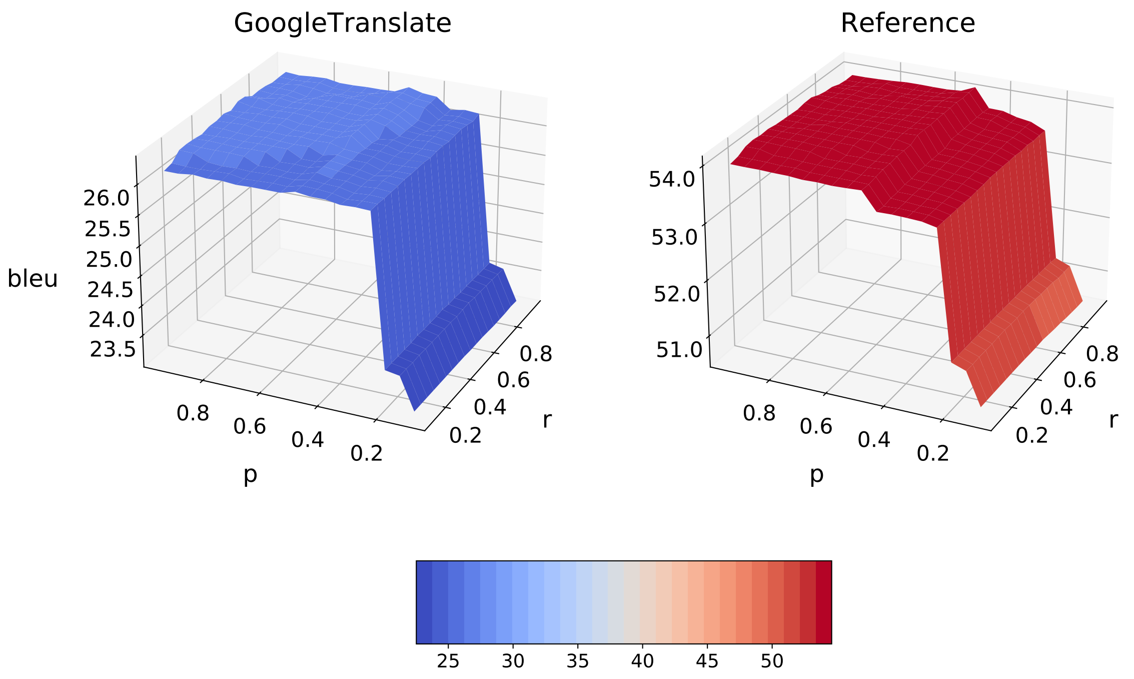


Figure A.3.: Evaluation of BLEU performance of oracle recovery for imperfect prefixes with a beam threshold of $b = 0.3$.

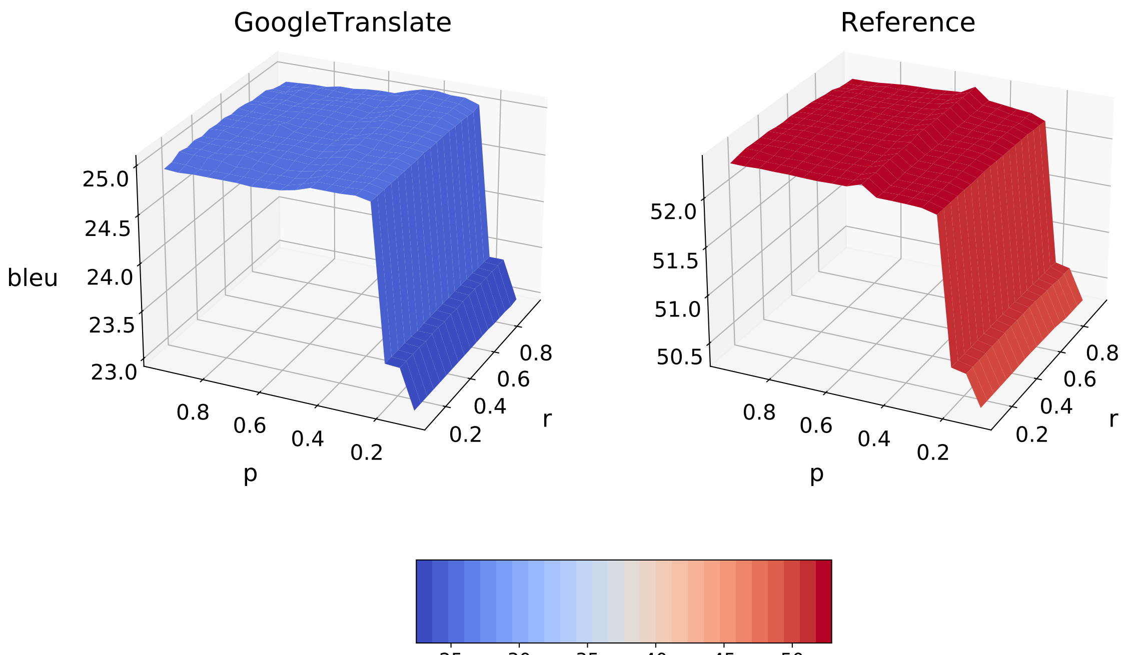


Figure A.4.: Evaluation of BLEU performance of oracle recovery for imperfect prefixes with a beam threshold of $b = 0.4$.