RUNTIME AUDIT OF NEURAL SEQUENCE MODELS FOR NLP

by

Shuoyang Ding

A dissertation submitted to The Johns Hopkins University in conformity with the requirements for
the degree of Doctor of Philosophy.

Baltimore, Maryland

July 2022

© 2022 Shuoyang Ding

All rights reserved
Abstract

Neural network sequence models have become a fundamental building block for natural language processing (NLP) applications. However, with the increasing performance and widespread adoption of these models, the social effects caused by errors in these models’ outputs are also amplified. This thesis aims to mitigate such adverse effects by studying different methods that generate user-interpretable auxiliary signals along with model predictions, thus enabling efficient audits of the model output at runtime.

We will look at two different types of auxiliary signals respectively generated for the input and the output of the model. The first type explains which input tokens are important for a certain prediction (Chapter 3 and 4), while the second estimates the quality of each output token (Chapter 5 and 6). For model explanations, our focus is to establish a comprehensive and quantitative evaluation framework, thus enabling a systematic comparison of different model explanation methods on a diverse set of architectures and configurations. For quality estimations, because there is already a solid evaluation framework in place, we instead focus
ABSTRACT

on improving state of the art by introducing an end-task-oriented pre-training step that is based on a non-autoregressive neural machine translation architecture. Overall, we show that it is possible to generate auxiliary signals of high quality with little to no human supervision, and we also provide some guidance for best practices regarding future applications of these methods to NLP, such as conducting comprehensive quantitative evaluations for the auxiliary signals before deployment, and selecting the appropriate evaluation metric that best suits the user’s goal.

Primary Reader and Advisor: Philipp Koehn

Secondary Readers: Marcin Junczys-Dowmunt, Anqi Liu
Acknowledgments

First and foremost, this thesis would not have been completed without the guidance and support from my PhD advisor, Philipp Koehn. Throughout my years of working with him, Philipp had always been an example of a fine empiricist to me, an objective I am still striving to attain. He gave me ample room to explore autonomously but was also ready to throw a branch in case I were up the creek without a paddle. He also genuinely cared about creating opportunities for his students to grow (maybe me especially as his first PhD student recruited through JHU) – from my experience, proposing a visit to his previous research group at The University of Edinburgh, encouraging me to do research internships at various industry labs, and so on. His symbolic “nothing fazes this dude”\(^1\) attitude had a lasting impact both in and out of my academic life. I owe him a debt of gratitude for all he had done for me as my PhD advisor.

I would also like to thank my thesis committee: Marcin Junczys-Dowmunt and Anqi

\(^1\)Credit to [https://github.com/moses-smt/mosesdecoder/blob/f8f4087aac52a5686f2a5c7a498dbba670919165/moses/Parameter.cpp#L1534](https://github.com/moses-smt/mosesdecoder/blob/f8f4087aac52a5686f2a5c7a498dbba670919165/moses/Parameter.cpp#L1534)
ACKNOWLEDGMENTS

Liu. Marcin also happened to be the host both for an invited talk and a prolonged internship of mine at Microsoft between 2019 and 2021. He was inspiring and whimsical enough to turn the most dreadful starting months of the COVID-19 pandemic into a segment of fond memory (even being trapped in a 270 sq ft compact studio in Seattle most of the time). Anqi was helpful throughout the whole writing process, especially with her invaluable perspectives as a Trustworthy AI researcher and suggestions to improve the coherence of the thesis. I also greatly appreciate their flexibility in holding the first in-person thesis defense at CLSP after the onset of the COVID-19 pandemic, making the defense a much more genial and memorable experience than yet another zoom meeting on the screen.

During my PhD years, CLSP had been my academic home. I was incredibly fortunate to learn and collaborate with a world-class team of professors, including Dan Povey, Shinji Watanabe, and Sanjeev Khudanpur. Among them, I would like to specifically thank Kevin Duh and Matt Post, who at different stages supervised me on various projects. Kevin could consistently ask thought-provoking questions at seminars and during our meetings, and was always warm-hearted enough to provide useful feedback on my papers and talks, as well as excellent career advice. Matt always had the kind of super-knowledgeable yet super-approachable post-doc researcher vibe (although he was well past that career stage even when I started). I could always learn a few new things after sitting down with him either working on some code or talking through a research idea.
ACKNOWLEDGMENTS

Life without my lovely colleagues at CLSP would be dull and bland. I would like to thank Tongfei Chen for sharing an apartment with me for six years, and for the myriad of intellectual conversations we had on research, coding, and random things in life; Hainan Xu and Winston Wu for the fun time we spent with the baby grand pianos in Mattin Center; Adi Renduchintala and Matthew Wiesner for grabbing post-seminar caffeine fix with me at Levering Hall or Carma’s, and for the brainstorming conversations we had on the way (some of which eventually turned into papers); Mo Yu, Nanyun Peng, and Yuan Cao for regularly checking in on me and helping me navigate my PhD life during my first few years; Jeremy Gwinnup, Huda Khayrallah, Rebecca Knowles, Gaurav Kumar, Xutai Ma, Kelly Marchisio, Becky Marvin, Arya McCarthy, Adi Renduchintala, Liz Salesky, Pamela Shapiro, Brian Thompson, Neha Verma, Rachel Wicks, and Xuan Zhang for being awesome colleagues in the machine translation group; And there were also Nanxin Chen, Mitchell Gordon, Biman Gujral, Pengyu He, Jonathan Jones, Yash Kumar Lal, Ke Li, Chu-Cheng Lin, Hongyuan Mei, Sabrina Mielke, Aaron Mueller, Adam Poliak, Roger Que, Pushpendre Rastogi, Rachel Rudinger, Keisuke Sakaguchi, Suzanna Sia, David Snyder, Dingquan Wang, Yiming Wang, Zach Wood-Doughty, Minhua Wu, Patrick Xia, Jinyi Yang, Sheng Zhang, and many others. You all have made CLSP the inclusive and supportive community as it is, and I have learned so much by interacting and collaborating with you.

Big thanks to the CLSP administration staff: Ruth Scally, Yamese Diggs, Carl Pupa,
ACKNOWLEDGMENTS

Jennifer Linton, and Lauren Bigham, and also administrators at the CS Department: Zachary Burwell and Kim Franklin. These are the people who work hard to keep everything running smoothly and provide invaluable administrative assistance, which often saves me days of stress (especially as an international student).

Outside JHU, I had the opportunity to visit The University of Edinburgh, Amazon, Salesforce, and Microsoft either as an intern or as a researcher. I would like to thank my hosts for these visits, including Rico Sennrich, Phil Williams, Matthias Huck, Alon Lavie, Michael Denkowski, Greg Hanneman, Ann Clifton, James Bradbury, Caiming Xiong, Marcin Junczys-Dowmunt, Christian Federmann, and Arul Menezes. Among them, I want to give a special thanks to Alon Lavie, for his sustained research and career advice long after the internship, and for picking a great email alias for me at Amazon (which I reclaimed recently as a full-time employee). Apart from these people, I would also like to thank my undergraduate thesis advisor Weiwei Sun from Peking University (now at the University of Cambridge), for offering me the opportunity to work on cutting-edge research problems as an undergrad, and for advising me on graduate school applications. To this day, I still look up to her as an inspiring role model who is always ready to take on the most fundamental and challenging problems in research.

I would like to thank my friends outside my professional circle: Jiangfan Deng, Chen Jiang, Tong Lian, Zhouxing Lu, Yu Tian, Yilin Xiong, and many others, for riding the ups
ACKNOWLEDGMENTS

and downs with me and always being there as empathetic listeners.

Finally, I would like to thank my family: my girlfriend Fangyuan Yan, for supporting me through the uncertain final leg of this journey with love and company; my parents, for always encouraging me to pursue whatever course my passion lies in and consistently serving as my exemplars of kindness, integrity, and professionalism. I love you all.
此中有真意，欲辨已忘言。
Contents

Abstract ii

Acknowledgments iv

List of Tables xvi

List of Figures xix

1 Introduction 1

2 Background 7

2.1 Runtime Model Audit 7

2.1.1 Concept 7

2.1.2 Methodology 13

2.2 Sequence Models for NLP 16

2.2.1 Sequence Models without Sequence-Level Input Context 17
CONTENTS

2.2.2 Sequence Models with Sequence-Level Input Context . . . . . . . 21

2.3 Explanation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 25

2.3.1 Problem Definition . . . . . . . . . . . . . . . . . . . . . . . . . . 25

2.3.2 Methods of Explanations . . . . . . . . . . . . . . . . . . . . . . 30

2.3.3 Evaluation of Explanations . . . . . . . . . . . . . . . . . . . . . 40

2.4 Quality . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 45

2.4.1 Problem Definition . . . . . . . . . . . . . . . . . . . . . . . . . . 45

2.4.2 Methods of Quality Estimation . . . . . . . . . . . . . . . . . . . 46

2.4.3 Evaluation of Quality Estimation . . . . . . . . . . . . . . . . . . 58

2.5 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 62

I Explanation 65

3 A Framework for Evaluating Explanations – Case Study on Neural Language Models 67

3.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 67

3.2 Evaluation Paradigm . . . . . . . . . . . . . . . . . . . . . . . . . . . 70

3.2.1 Plausibility . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 70

3.2.2 Faithfulness . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 73

3.3 Data . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 75
CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.4 Composition Scheme</td>
<td>78</td>
</tr>
<tr>
<td>3.5 Experiments</td>
<td>82</td>
</tr>
<tr>
<td>3.5.1 Setup</td>
<td>82</td>
</tr>
<tr>
<td>3.5.2 Main Results</td>
<td>85</td>
</tr>
<tr>
<td>3.5.3 Analysis</td>
<td>88</td>
</tr>
<tr>
<td>3.6 Discussion</td>
<td>95</td>
</tr>
<tr>
<td>3.7 Conclusion</td>
<td>97</td>
</tr>
<tr>
<td>4 Comparison of Explanations from Different Methods – Case Study on Neural Machine Translation</td>
<td>99</td>
</tr>
<tr>
<td>4.1 Introduction</td>
<td>99</td>
</tr>
<tr>
<td>4.2 Methods</td>
<td>102</td>
</tr>
<tr>
<td>4.2.1 Attention</td>
<td>102</td>
</tr>
<tr>
<td>4.2.2 Word Alignment Models</td>
<td>104</td>
</tr>
<tr>
<td>4.2.3 Gradient-based Post-hoc Explanations</td>
<td>106</td>
</tr>
<tr>
<td>4.3 Experiments</td>
<td>106</td>
</tr>
<tr>
<td>4.3.1 Evaluation Method</td>
<td>106</td>
</tr>
<tr>
<td>4.3.2 Setup</td>
<td>108</td>
</tr>
<tr>
<td>4.3.3 Force Decoding Results</td>
<td>111</td>
</tr>
<tr>
<td>4.3.4 Free Decoding Results</td>
<td>114</td>
</tr>
</tbody>
</table>
## 6.2 Levenshtein Training for Word-level Quality Estimation

6.2.1 Motivation ............................................. 141
6.2.2 Levenshtein Transformer ................................. 143
6.2.3 Pre-trained Model ................................... 145

## 6.3 Transfer Learning from Translation to Word-level Quality Estimation

6.3.1 Synthetic Data Construction ......................... 147
6.3.2 Compatibility with Subwords ......................... 149
6.3.3 Label Imbalance .................................. 152
6.3.4 Ensemble ...................................... 153

## 6.4 Experiments on WMT 2020 Dataset

6.4.1 Data Setup .................................... 155
6.4.2 Model Setup .................................... 158
6.4.3 Evaluation Setup ................................ 159
6.4.4 Results ....................................... 159

## 6.5 Experiments on WMT 2021 Dataset

6.5.1 Data Setup .................................... 163
6.5.2 Model Setup .................................... 165
6.5.3 Devtest Results ................................ 166
6.5.4 Analysis ....................................... 169
List of Tables

2.1 Summary of the methods and evaluations we have covered in Section 2.3 and Section 2.4 ................................................................. 63

3.1 An example from our evaluation where different saliency methods assign different importance scores for the same model (Transformer language model) and the same next word prediction (are). G, SG and IG stands for gradient Saliency method, SmoothGrad and Integrated Gradients, respectively (see Section 2.3.2 for details). The tints of green and yellow mark the magnitude of positive and negative importance scores, respectively. .......................... 68

3.2 Examples prefixes from the four evaluation datasets, followed by the probing tag prediction under the expected scenario. The cue and attractor sets are marked with solid green and yellow, respectively. .................................................. 76

3.3 Plausibility case study result. Each number is the fraction of cases the explanation passes the benchmark test, while the numbers in brackets for each architecture are the fraction of times these scenarios occur for predictions generated by the corresponding model. Results from the best explanation method for each architecture are boldfaced. The exp. and alt. columns are breakdown of evaluation results into expected scenarios and alternative scenarios as defined in Section 3.2. G, SG, IG stands for the gradient saliency, SmoothGrad, and Integrated Gradients, respectively. .................................... 85

3.4 Faithfulness Benchmark Result. Each number is the average Pearson correlation computed on the corresponding dataset. Results from the best explanation method for each architecture are boldfaced. Refer to the caption of Table 3.3 for other notations. ................................................................. 86

3.5 Plausibility benchmark result for Vector Norm (VN) composition scheme. Refer to the caption of Table 3.3 for notations. ........................................ 89
### LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.6</td>
<td>Examples from Winobias dataset for qualitative analysis. Cue words are marked with [ ] while attractor words are marked with ( ). The tints of <strong>green</strong> and <strong>yellow</strong> mark the magnitude of positive and negative importance scores, respectively. For all examples, the prediction interpreted is the <strong>FEMININE</strong> tag. 1 is a case with high plausibility and low input faithfulness; 2 is a case with low plausibility and high input faithfulness; 3 is a case with low model faithfulness; 4 is a case with high plausibility and high input/model faithfulness.</td>
</tr>
<tr>
<td>3.7</td>
<td>Plausibility &amp; input faithfulness on synthetic datasets with distilled models. Only results for the interpretation method with best performance are shown. Refer to the caption of Table 3.3 for other notations.</td>
</tr>
<tr>
<td>3.8</td>
<td>A number agreement test case where the distilled Transformer model makes the correct prediction (singular) but all explanation methods unanimously point to a singular noun that is not grammatical subject as the most salient cue for this prediction.</td>
</tr>
<tr>
<td>4.1</td>
<td>Alignment Error Rate (AER) with different explanation methods, under <strong>force decoding</strong> setting. GIZA++ and fast-align Offline results are quoted from Zenkel, Wuebker, and DeNero (2019), whereas fast-align Online stands for our online alignment result (c.f. Section 4.3.2). <strong>bidir</strong> refers to the symmetrized alignment results. Best results for each architecture are marked with underlines, and best explanation/alignment results are respectively marked with boldface. Numbers affected by hyper-parameter tuning are marked with *.</td>
</tr>
<tr>
<td>4.2</td>
<td>Alignment Error Rate (AER) with different explanation models, under <strong>free decoding</strong> setting. See the caption of Table 4.1 for notations.</td>
</tr>
<tr>
<td>4.3</td>
<td>Alignment distribution entropy for selected de-en models.</td>
</tr>
<tr>
<td>4.4</td>
<td>Stability analysis of results with multiple runs in Table 4.1. Numbers affected by hyper-parameter tuning are marked with *.</td>
</tr>
<tr>
<td>5.1</td>
<td>Comparison of Matthews Correlation Coefficient on translation words (word MCC) between dedicated Predictor-Estimator (PredEst) quality estimation model, quality estimations built from translation probability (TP), and quality estimations built from Monte-Carlo Dropout (MCD).</td>
</tr>
<tr>
<td>6.2</td>
<td>Constrained setting. All LevT models here are transformer-base models. F1-OK and F1-BAD are F1 scores of the OK and BAD tags, respectively. Higher is better for all metrics in this table.</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>Table Number</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.3</td>
<td>Unconstrained setting. base and big stand for the transformer-base and</td>
<td>160</td>
</tr>
<tr>
<td></td>
<td>transformer-big architecture. 418M is the M2M-100-small model.</td>
<td></td>
</tr>
<tr>
<td>6.4</td>
<td>Ablation analysis. All results trained with synthetic data in this table use</td>
<td>162</td>
</tr>
<tr>
<td></td>
<td>the src-mt1-mt2 data synthesis method. +lang-adapt stands for adding</td>
<td></td>
</tr>
<tr>
<td></td>
<td>an extra autoregressive MT training step using the same parallel training</td>
<td></td>
</tr>
<tr>
<td></td>
<td>data as LevT training, so the M2M-100 model is adapted to translating a</td>
<td></td>
</tr>
<tr>
<td></td>
<td>specific language pair.</td>
<td></td>
</tr>
<tr>
<td>6.5</td>
<td>Data Source and Statistics of Parallel Datasets Used in Our WMT 2021</td>
<td>163</td>
</tr>
<tr>
<td></td>
<td>Experiments</td>
<td></td>
</tr>
<tr>
<td>6.6</td>
<td>Target MCC results on test20-v2 dataset for all language pairs we submitted</td>
<td>167</td>
</tr>
<tr>
<td></td>
<td>systems for. Stage 2 stands for synthetic finetuning (where N stands for</td>
<td></td>
</tr>
<tr>
<td></td>
<td>not performing this stage). Stage 3 stands for human annotation finetuning.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>µ stands for the label balancing factor.</td>
<td></td>
</tr>
<tr>
<td>6.7</td>
<td>Analysis of different data synthesis methods on en-de language pair. All</td>
<td>168</td>
</tr>
<tr>
<td></td>
<td>models here are initialized with M2M-100-small.</td>
<td></td>
</tr>
<tr>
<td>6.8</td>
<td>Analysis of src-mt1-mt2 and mvpe method on ro-en and et-en language pair.</td>
<td>168</td>
</tr>
<tr>
<td>6.9</td>
<td>Analysis of different label balancing factors initialized on to-English</td>
<td>170</td>
</tr>
<tr>
<td></td>
<td>language pairs. All results are based on the multilingual model and not</td>
<td></td>
</tr>
<tr>
<td></td>
<td>performing synthetic finetuning step.</td>
<td></td>
</tr>
<tr>
<td>6.10</td>
<td>Detailed evaluation metric breakdown of all submitted ensemble system on</td>
<td>171</td>
</tr>
<tr>
<td></td>
<td>test20 test set.</td>
<td></td>
</tr>
<tr>
<td>A.1</td>
<td>Parameter size (in millions) and perplexity on WikiText-103 dev set for all</td>
<td>188</td>
</tr>
<tr>
<td></td>
<td>language models we trained.</td>
<td></td>
</tr>
<tr>
<td>A.2</td>
<td>Addition interpretation examples with LSTM.</td>
<td>189</td>
</tr>
<tr>
<td>A.3</td>
<td>Addition interpretation examples with QRNN.</td>
<td>190</td>
</tr>
<tr>
<td>A.4</td>
<td>Addition interpretation examples with Transformer.</td>
<td>191</td>
</tr>
</tbody>
</table>
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>A summary of the encoder-decoder architecture.</td>
<td>26</td>
</tr>
<tr>
<td>2.2</td>
<td>An illustrative example for ROC curve. Figure from Wikipedia contributors (2022).</td>
<td>60</td>
</tr>
<tr>
<td>3.1</td>
<td>Illustration of language model prediction head replacement for model explanation evaluation. All language models are first trained with the word prediction head following the standard language model training recipe, then a probe prediction head is finetuned on synthetic finetuning data and used for evaluations of model explanation methods.</td>
<td>72</td>
</tr>
<tr>
<td>3.2</td>
<td>Analysis of model configuration vs. plausibility on PTB and CoNLL benchmark. Each model configuration is color-coded, while the parameter size (in millions) is shown with circle size. l, w, e, h stands for model depth, width of feed-forward layers after self-attention, embedding size, and the number of heads.</td>
<td>92</td>
</tr>
<tr>
<td>3.3</td>
<td>Analysis of model configuration vs. plausibility and faithfulness on Syneval benchmark. Each model configuration is color-coded, while the parameter size (in millions) is shown with circle size. l, w, e, h stands for model depth, width of feed-forward layers after self-attention, embedding size, and the number of heads. Note that the faithfulness numbers plotted here are the ones explained with expected scenario predictions.</td>
<td>93</td>
</tr>
<tr>
<td>4.1</td>
<td>Comparison of our gradient-based word alignment explanation of convolutional NMT model with reference and attention explanation.</td>
<td>101</td>
</tr>
<tr>
<td>4.2</td>
<td>SmoothGrad explanation of FConv de-en Model with VN and GI composition schemes.</td>
<td>116</td>
</tr>
<tr>
<td>4.3</td>
<td>Explanation of Transformer de-en model with different SmoothGrad noise values ( \sigma (n = 30) ).</td>
<td>118</td>
</tr>
</tbody>
</table>
6.1 Figure (a) shows an example of TER-style edit labels used as reference for word-level quality estimation task. Figure (b) shows a series of hypothetical Levenshtein Transformer edits that generates the same sequence from the target input. The similarity of these edit operations motivates the study in this paper. .............................. 144
Chapter 1

Introduction

Human language presents itself as sequences of symbols. Hence, the modeling of sequences is one of the most fundamental problems of natural language processing (NLP). In recent years, with the innovative use of neural networks, these sequence models have achieved a high level of performance both for understanding and generating languages. For example, finetuned neural masked language models have surpassed human performance for both GLEU and SuperGLEU benchmarks for natural language understanding (Wang et al., 2019b; Wang et al., 2019a). On the generation side, neural machine translation (NMT) systems have also reached an impressive level of quality in some settings, which starts to trigger discussions of human parity (Hassan et al., 2018; Läubli, Sennrich, and Volk, 2018).

However, these results do not indicate that these neural sequence models are infallible,
as real-world performance tends to be overestimated with held-out datasets (Ribeiro et al., 2020). With improving performance and wider adoptions, the social effects caused by the errors and pathologies of these neural sequence models are also more amplified. In light of these effects, there have been multiple calls for building Trustworthy Artificial Intelligence (AI) systems (Liu et al., 2021; Li et al., 2021) that put the concept of auditability as one of its core requirements. For an AI system to be auditable, it “requires that the justification of a system to be reviewed, assessed, and audited” (Liu et al., 2021).

Throughout the life cycle of an AI system, audits could happen at any stage, on any aspect of the system-building process. For example, at the design stage, the data source can be audited to ensure proper domain coverage for the use case. At the development stage, the behavior of the system can be audited to ensure that it adheres to the social and ethical values of its stakeholders. At the deployment stage, the deployment pipeline can be audited to ensure that its speed benchmark aligns with the workload the system will process at runtime. All these audits happening at different stages are crucial for building an AI system with as few adverse impacts on the users as possible.

But there is another inevitable aspect of this scenario that we have to face: what if our deployed AI system still produces some errors during interactions with users, after all these internal audits? This is an especially important problem for models that generate sequences because fluent but inaccurate outputs have been observed in several different
CHAPTER 1. INTRODUCTION

applications, including machine translation (Bentivogli et al., 2016; Castilho et al., 2017; Raunak, Menezes, and Junczys-Dowmunt, 2021), abstractive summarization (Zhao, Cohen, and Webber, 2020; Durmus, He, and Diab, 2020), and image captioning (Rohrbach et al., 2018). Some of these outputs may also be perceived as toxic (Gehman et al., 2020; Specia et al., 2020b) or biased (Sun et al., 2019). Even occasional, they may cause significant adverse effects to the users of these systems.

This scenario leads us to audits at the runtime stage after the development and deployment are finalized. Compared to other stages of the AI system life cycle, the model audit that happens at this stage has the following characteristics:

1. The audit will happen through interactions with users, without the participation of system developers.

2. The audit will focus on notifying the users of instance-level anomaly behavior and assume that any global-level behavior has already been thoroughly investigated during the system development stage.

3. The audit will be conducted without reference output, unlike how system evaluation is normally done during the model development stage.

Only the argmax predictions made by the model are not a lot of information for such an audit. On the other hand, in theory, we could grant the users full transparency to the
CHAPTER 1. INTRODUCTION

inference process, but due to the obscurity of the neural network models, it is overwhelming for the users to parse all the intermediate activations of the model at once. To effectively audit the model, the users will need auxiliary signals apart from model predictions in order to locate and diagnose problems. Hence, the central question addressed by this thesis is the following:

How do we generate user-interpretable auxiliary signals for the runtime audit of neural sequence models in the context of NLP?

Like model audits happening during other stages, there are multiple ways model audits can be performed, thus creating different auxiliary signals. In this thesis, we focus on tackling the following two types of auxiliary signals:

- For a model with sequential inputs, can we explain which input tokens are important for a certain prediction?

- For a model with sequential outputs, can we estimate the quality of each output token?

Methods that generate these two different types of signals are quite different but can be categorized similarly: the intrinsic methods and post-hoc methods. The goal of intrinsic methods is to build models that can generate relevant auxiliary signals as an intermediate step during inference, while the goal of post-hoc methods is to run specific signal-generating procedures after the inference. Our study in this thesis covers a diverse set of methods from
both categories for both types of auxiliary signals, with stress on quantitative evaluation and realistic models.

Within this problem context, this thesis mainly seeks to make contributions in the following aspects:

- For the auxiliary signals of explanation, many methods that generate these signals were proposed in the context of computer vision. Hence, we identify two main challenges regarding deployment these methods to NLP: (1) properly composing neuron-level feature importance to token-level importance score (Section 3.4), and (2) designing paradigms to quantitatively evaluate the auxiliary signals (Section 3.2 and 4.3.1). Our proposals are empirically investigated in the context of two NLP sequence modeling tasks (Section 3.5 and 4.3).

- For the auxiliary signals of quality estimation, we focus on determining the optimal approach for generating such signals through comparative analysis (Chapter 5) and further improving the accuracy of signals by advancing the state-of-the-art approach (Chapter 6).

Below we give a brief summarization of the remaining chapters in this thesis:

- Chapter 2 introduces the background of our work, including the concept of runtime model audit, neural sequence models, and methods for generating auxiliary signals.
CHAPTER 1. INTRODUCTION

- Chapter 3 establishes a framework for evaluating model explanations in the context of neural language models (published as Ding, Xu, and Koehn (2019)).

- Chapter 4 systematically compares model explanations in the context of neural machine translation (NMT) models (published as Ding and Koehn (2021)).

- Chapter 5 systematically compares the word-level quality estimation in the context of NMT models.

- Chapter 6 improves current quality estimation practices by treating the training procedure of a non-autoregressive machine translation architecture as a pre-training step targeted for the word-level quality estimation task (published as Ding et al. (2021b) and Ding et al. (2021a)).

- Chapter 7 summarizes the main findings in this thesis and points to some directions for future work.
Chapter 2

Background

2.1 Runtime Model Audit

2.1.1 Concept

As we mentioned in the previous chapter, an AI system with auditability “requires that the justification of a system to be reviewed, assessed, and audited” (Liu et al., 2021). Since model auditing is a broad concept not often discussed in the context of NLP, we will focus on establishing connections with existing work in NLP by looking at the life cycle of the machine translation system as an example, thus also explaining how runtime audit is different from the other stages. Then we switch our discussion to the auxiliary signals that
CHAPTER 2. BACKGROUND

help perform runtime audits, namely why they are necessary, and why we choose to focus on explanation and quality. Note that our discussion will be constrained within the research and development facet of a system. While we acknowledge that informed management and policy are critical aspects of building trustworthy AI systems, this thesis is not in a position to discuss those issues.

We divide the life cycle of an AI system into four stages: design, development, deployment, and runtime. This is inspired by the three-stage presentation from Silva and Alahakoon (2021) but designates the original “Monitor and Evaluate Performance” step during the deployment as a stand-alone stage.

Example: Audit on Machine Translation System

We now go over each stage of this life cycle for a machine translation system and discuss what current work constitutes audit for different stages.

Design At this stage, the existing literature is reviewed to make decisions on the architecture and algorithm. Data acquisition and preparation also happen at this stage. In the machine translation context, most of the work that analyzes and justifies a specific architecture design can be viewed as audits of this stage. Some examples for this include investigations on whether to use length normalization or reward (Murray and Chiang, 2018), on whether to use non-autoregressive models (Kasai et al., 2021), and on whether to integrate visual
Investigations on whether pre-trained models are helpful for machine translation (Zhu et al., 2020; Yang et al., 2020; Xu, Van Durme, and Murray, 2021) also fall into this category. Furthermore, there are also efforts on auditing the quality of the web-crawled corpus (Caswell et al., 2021), how different kinds of noises in the data affect system performance (Khayrallah and Koehn, 2018), and whether a system violates data provenance (Song and Shmatikov, 2019; Hisamoto, Post, and Duh, 2020). These studies all provide justifications for data-related decisions in the machine translation system-building process.

**Development** At this stage, the focus is on specific techniques and configurations that transform prepared data and designed algorithms into executable models, and how to evaluate those models. Examples for audit at this stage include investigations on the number of merge operations used for segmenting words into subwords (Ding, Renduchintala, and Duh, 2019) and on the usage of perturbation-based regularization techniques (Takase and Kiyono, 2021). There are also a lot of studies on auditing the machine translation evaluation process, like what metrics correlate the best with human judgments (Kocmi et al., 2021) and what references to use for evaluation (Freitag et al., 2020).

It should be pointed out that there is not always a clear cut between the audit of the design vs. development stage, as some audits are driven by a particular phenomenon rather than focusing on a specific stage of the life cycle. For example, there are a lot of recent
studies on auditing gender bias (Stanovsky, Smith, and Zettlemoyer, 2019; Savoldi et al., 2021) and toxic output (Specia et al., 2020b; Wang et al., 2021) in machine translation, which will end up examining practices either in the design or development stage depending the root cause identified.

**Deployment**  At this stage, the model will be transferred from the development environment to the production environment. Secondary metrics such as efficiency and ability to handle high concurrency will also be examined at this stage. Audits at this stage are rarely documented as research papers as most of the effort concentrates on engineering. The only example of audit at this stage is the studies that examine quality-efficiency trade-offs of machine translation systems (Heafield, Zhu, and Grundkiewicz, 2021).

**Runtime**  At this stage, the system has finished all steps of the system-building process and entered production. Updates to the model are limited, and the system will interact with the users instead of researchers and engineers. Hence, at this stage, the goal of the audit is no longer to identify ways to improve the system, but rather to notify the users of erroneous behaviors that may potentially have adverse effects, and support any post-hoc fixes if applicable. Besides, since we don’t know what input will feed into the system, the audit will also need to be performed without references, which is also different from many audits in previous stages.
CHAPTER 2. BACKGROUND

**Auxiliary Signal for Runtime Audit**

Now that we have defined the concept of runtime audit of an AI system, we now define the concept of *auxiliary signal*, which is the center of *how* to conduct such audit.

Based on our definition of the task, we cannot expect the users to detect erroneous behaviors based only on the input/output pairs from the AI system, because (1) human users cannot process a large amount of data as efficiently as an AI system would do, and (2) users may not even have the necessary specialized knowledge to judge the output of the system. We could also, in theory, grant the users full transparency and allow them to look at the implementation and parameters of the model. Such transparency would be helpful when the model is inherently easy to understand, but since the discussion in this thesis is mostly concerned with neural network models, such transparency would violate the principle of avoiding cognitive overload, which was proposed in several previous studies on interactions between human and AI systems (Kulesza et al., 2013; Eiband et al., 2018). Hence, it is critical to strike a balance by providing users with the right amount and format of signals that (1) help them quickly locate the erroneous examples, and (2) help them understand and fix the problem without creating any cognitive overload. In the rest of the thesis, we will refer to such signal as *auxiliary signal*.

For the first purpose, an automated *quality estimation* is the most direct signal for identifying anomaly outputs. Such estimation may come as continuous scores or binary
CHAPTER 2. BACKGROUND

labels and may be generated either at the word level or sequence level.

For the second purpose, both word-level and sequence-level quality estimation have also been shown to help human workers collaborate more efficiently with AI systems for applications that involve human post-editing or human-in-the-loop inference (Turchi, Negri, and Federico, 2015; Knowles and Koehn, 2018; Lee et al., 2021). Apart from quality estimation, an explanation for the decision also serves as a useful signal to the users. Studies in psychology have pointed out that people tend to ask for explanations to improve their understanding (Heider, 2013), especially when encountering anomaly phenomena (Hilton and Slugoski, 1986; Hesslow, 1988; Hilton, 1996). In the field of explainable AI (XAI), explanations have been described as providing insights to the system users (Kim and Doshi-Velez, 2021), improving their decision making (Wang et al., 2019c), and improving trust of the users in the AI system (Jacovi et al., 2021). From a pragmatic perspective centered around sequence modeling, explanations that come in the form of word alignments could help detect hallucination and under-generation errors (Zhao et al., 2019; Zhou et al., 2021), which means the model is either generating something that is not prompted in the input context, or the model is missing out on some critical information, respectively. Feature importance explanations can also help the users perform rule-based checks, e.g., morphosyntactic well-formedness (Pratapa et al., 2021). Compared to the estimate of quality, explanations of model predictions can come in even more forms, which
CHAPTER 2. BACKGROUND

we will discuss in detail in Section 2.3.

There are many other auxiliary signals that we can generate at runtime beyond quality estimation and explanation, such as membership inference (Hisamoto, Post, and Duh, 2020, for privacy audit), input perturbations (Niu et al., 2020, for robustness audit), and so on. However, keep in mind that the primary goal of runtime audits is to detect anomaly behavior, and audits for aspects such as privacy and robustness should already be conducted during the system development stage. Hence, to avoid cognitive overload for the users, unless there are some application-specific requirements, those signals should not be included for runtime model audit in general.

2.1.2 Methodology

In this section, we will talk about several different categorizations of methods that can be used to generate the auxiliary signals needed for runtime model audit. We will be largely following some of the common categorization in model explainability literature (Lipton, 2018; Dosilovic, Brcic, and Hlupic, 2018; Arrieta et al., 2020; Bodria et al., 2021; Guidotti et al., 2019; Molnar, 2019), but the readers shall see in the later sections that such categorization will also apply for the problem of quality estimation.

The most important distinction we make is whether the auxiliary signals come from the intrinsic intermediate steps during inference (intrinsic), or come from signals generated
after the inference has completed (post-hoc).

In the first case, either model weights or specific intermediate model activations that are deemed as interpretable are shared with the users for audit. Often, the effort of designing a method in this case is to design a module that has inherently interpretable model activations, so no further step is necessary to generate the auxiliary signal required for model audit. There are some downsides to this approach. Because this essentially adds another constraint beyond optimal task performance, the model developer needs to ensure that the interpretable model does not adversely affect the inference performance or speed. Also, there are often some doubts on whether the claims about the interpretability of some modules are really reliable (Adebayo et al., 2018; Kindermans et al., 2019). We will also study this problem under the context of neural machine translation in Chapter 4.

On the other hand, the post-hoc methods can be further broken down into two different categories. To generate the auxiliary signal, we use either the probability, gradient, or pseudo-gradient generated by certain back-propagation rules, all based on the original inference model (original), or a dedicated model that is separate from the original inference model is used to generate the auxiliary signal (dedicated). The trade-off between the two methods is straightforward: the first approach is less expensive to run and more faithful to the decision mechanism original model, but could be noisy at times; the second approach often generates higher-quality signals, but it either requires some unsupervised method
CHAPTER 2. BACKGROUND

to generate the signal or requires training data with human-annotated signals to train the model. We will repeatedly revisit this trade-off at the later parts of the thesis.

Another distinction that is often made is whether the audit happens at the global or local level. Global methods examine the decision of a model as a whole and provide insights on the scale of at least a batch of data instances. Normally they will focus on revealing certain decision criteria of the model or try to understand what information is and/or is not captured by the model. Examples of such methods include Decision Trees, Concept Activation Vectors (Kim et al., 2018, CAV), probing (Shi, Padhi, and Knight, 2016; Adi et al., 2017; Conneau et al., 2018; Tenney, Das, and Pavlick, 2019; Tenney et al., 2019; Kim et al., 2019; Vulić et al., 2020; Hall Maudslay and Cotterell, 2021), targeted challenge sets (Isabelle, Cherry, and Foster, 2017; Khashabi et al., 2018; Jumelet, Zuidema, and Hupkes, 2019). On the other hand, local methods examine the decision of a model on the level of a single instance. Because of the characteristics of runtime model auditing, we will focus on the local-level methods.

So far, our introduction is largely conceptual and does not get into the details of any specific method. We will introduce them separately in the context of explanation and quality estimation.
CHAPTER 2. BACKGROUND

2.2 Sequence Models for NLP

Given a sequence of words $y = y_0 y_1 \ldots y_J$ sampled from a given language $\mathcal{L}$ and sequence-level input context $x$, a sequence model estimates the probability of the sequence in the language space $\mathcal{L}$ conditioned on the input context $p(y \mid x)$. A common technique to estimate this probability mass is to factorize the probability into a sequence of conditional probabilities using the chain rule:

$$p(y \mid x) = p(y_0, y_1, \ldots, y_J \mid x)$$

$$= p(y_J \mid y_0, \ldots, y_{J-1}, x) \cdot p(y_0, \ldots, y_{J-1} \mid x)$$

$$= p(y_J \mid y_0, \ldots, y_{J-1}, x) \cdot p(y_{J-1} \mid y_0, \ldots, y_{J-2}, x) \cdot p(y_0, \ldots, y_{J-2} \mid x)$$

$$= \cdots$$

$$= p(y_J \mid y_0, \ldots, y_{J-1}, x) \cdot p(y_{J-1} \mid y_0, \ldots, y_{J-2}, x) \cdot \ldots \cdot p(y_1 \mid y_0, x) \cdot p(y_0 \mid x)$$

(2.1)

There are many ways such models can be instantiated, depending on the task of interest and model architecture of choice. The following subsections give an overview of how they are usually instantiated with or without the input context $x$, respectively. Architecture-wise, the scope of this thesis will focus on neural sequence models, i.e. using neural networks to build estimators for each individual component.
2.2.1 Sequence Models without Sequence-Level Input Context

We start by looking at the simpler case where no sequence-level input context is provided, i.e., \( x \) is an empty sequence. An example of such model in real-world applications is Language Model (LM)\(^1\). To facilitate presentation, we define the prefix of the first \( j \) input words as \( s_j = y_0y_1 \ldots y_{j-1} \). In that way, the output distribution of the \( j \)-th word \( p(y_j \mid y_0, y_1, \ldots, y_{j-1}) \) could be rewritten as \( p(y_j \mid s_j) \).

In a neural network, both the prefix \( s_j \) and the individual input word \( y_j \) will be represented as a vector representation \( h_j = \mathcal{H}(s_j) \) and \( e_j = \mathcal{E}(y_j) \). \( \mathcal{H} \) and \( \mathcal{E} \) are mappings from strings/words to their vector representations, respectively. Normally, \( \mathcal{E} \) is defined as a static mapping of a pre-defined vocabulary of words \( \mathcal{V} \) to their respective learnable parameter vectors. But since the space of \( s_j \) is exponentially growing with regard to the size of the vocabulary, we cannot afford to maintain such static mapping for \( \mathcal{H} \).

One popular solution is to build a **recurrent** model. Note that the prefix \( s_j \) could also be defined recursively as \( s_j = s_{j-1}y_{j-1} \). Utilizing this recursiveness, we can define the

---

\(^1\)Recent publications sometimes refer to Masked Language Model (Devlin et al., 2019, MLM) as Language Model, but such model does not have generation capabilities and does not conform with the definition of sequence models in this thesis. Throughout this thesis, unless otherwise stated, we use the term Language Model to refer specifically to the more traditional “left-to-right” Language Model.
mapping $\mathcal{H}$, or $h_j$, following the recursion:

$$h_j = f_r(h_{j-1}, e_{j-1}) \quad (2.2)$$

To complete the recursion, we need to define the base case $h_0$. In practice, $h_0$ is either defined as learnable parameters, or $h_0 = 0$.

With both $\mathcal{H}$ and $\mathcal{E}$ defined, we can build a neural network estimator for each factor in the sequence-level probability $p(y_j \mid s_j)$:

$$p(y_j \mid s_j) = f_o(\mathcal{H}(s_j)) = f_o(h_j) \quad (2.3)$$

The question is then reduced to how to define $f_r$ and $f_o$. There are many ways to do this. The simplest being Recurrent Neural Network (RNN) introduced by Mikolov et al. (2010):

$$h_j = \sigma(W^h(h_{j-1} + e_{j-1})) \quad (2.4)$$

$$p(y_j \mid s_j) = \frac{\exp(W^o_{y_j} h_j)}{\sum_{v \in \mathcal{V}} \exp(W^o_v h_j)} \quad (2.5)$$

where $\sigma$ is the sigmoid function. $W^h \in \mathbb{R}^{d_h \times d_h}$ and $W^o \in \mathbb{R}^{d_h \times |\mathcal{V}|}$ are both learnable parameters of the neural network. $d_h$ is the dimension of hidden states $h_j$. $W^o_v$ stands for the row that corresponds to the word $v$ in the vocabulary $\mathcal{V}$. With each factor defined, we
can compute the probability of a sequence \( p(y) \) following the factorization in Equation 2.1. Furthermore, we can train its parameters by maximizing the probability of a training corpus.

RNN is not the optimal way to define the \( f_r \) and \( f_o \) function above – specifically, it is known to suffer from the so-called exploding/vanishing gradient problem (Pascanu, Mikolov, and Bengio, 2013). Several different solutions such as Long Short-Term Memory (Hochreiter and Schmidhuber, 1997, LSTM) and Gated Recursive Unit (Cho et al., 2014, GRU) have been proposed to solve such problem, with the shared spirit of implementing a gating mechanism that prohibits the gradient from propagating indefinitely. Noting that the recursion will take \( O(J) \) time steps to fully unroll (\( J + 1 \) being the length of the sequence), there are several other efforts that build approximations of the recurrent units by either mixing shorter recursions with convolution and pooling (Bradbury et al., 2017), or building a pyramidal network of convolutions (Dauphin et al., 2017). Because the exact variants of \( f_r \) and \( f_o \) are not the focus of discussion in this thesis, we leave the readers to those individual papers for reference.

Up to this point, we have been building a recurrent model for the sequence modeling problem, thus assuming that all the context that is necessary to predict \( y_j \) is encoded in \( h_{j-1} \) and \( e_{j-1} \). A very popular non-recurrent solution, called self-attention (Vaswani et al., 2017), diverges significantly from this assumption. The key idea from self-attention is that instead of computing \( h_j \) from \( h_{j-1} \) and \( e_{j-1} \), we compute \( h_j \) from the individual
CHAPTER 2. BACKGROUND

embeddings of all the words in the prefix \( s_j \), i.e., \([e_0, \ldots, e_{j-1}]\) by taking a weighted sum of these individual states, while the summation weights are defined by a module that predicts how much “attention” we want to pay to each state\(^2\):

\[
h_j = \sum_{j'=0}^{j-1} a_{j'j} \mathbf{W}^v e_{j'} \tag{2.6}
\]

where \( \mathbf{W}^v \in \mathbb{R}^{d_e \times d_h} \) is a linear transformation. \( d_e \) is the dimension of the embeddings. \( a_{j'j} \) is attention weights that is computed with \( e_{j'} \) as the key and \( e_j \) as the query\(^4\), shown as below:

\[
a_{j'j} = \frac{\exp((\mathbf{W}^q e_j)^T \mathbf{W}^k e_{j'}/\sqrt{d_e})}{\sum_{l=1}^{j-1} \exp((\mathbf{W}^q e_j)^T \mathbf{W}^k e_l/\sqrt{d_e})} \tag{2.7}
\]

where \( \mathbf{W}^q, \mathbf{W}^k \in \mathbb{R}^{d_e \times d_h} \) are linear transformations corresponding to the query and the key, respectively. After defining \( h_j \), each factor distribution \( p(y_j | s_j) \) is defined similarly to the recurrent model as shown in Equation 2.5.

Self-attention is the most important building block of the Transformer model architecture

\(^2\)In accordance with the way we introduced RNN, we assume the network only has one layer, but note that this is almost never how self-attention networks are used in real applications. In the case when multiple layers are stacked, the presented computation will only apply for the first layer of the network, and the self-attention mechanism in the higher layers will instead attend to the hidden states of each individual position from the previous hidden layer.

\(^3\)Since we only focus on introducing high-level concepts, this formula omitted some details such as multi-head attention, residual connection and layer normalization for brevity. We recommend the readers check the original paper for full details of the model.

\(^4\)In accordance with this naming convention, the vectors \( e_{j'} \) used in the weighted sum computation in Equation 2.6 are called values in the literature.
(Vaswani et al., 2017), which has been used for a wide variety of applications including both language modeling (Radford et al., 2019; Brown et al., 2020) and neural machine translation (Vaswani et al., 2017). We will revisit Transformer and some of its variants repeatedly for the rest of this thesis.

2.2.2 Sequence Models with Sequence-Level Input Context

So far, we have been focusing on cases without sequence-level input context. However, the vast majority of NLP applications would require the output distribution to adapt to some input context. For example, when translating a sentence from a foreign language, the generated sentence should have the same semantics as the input sentence. When generating responses for a conversation agent, the generated sentence should fit into the conversation context and appropriately respond to the query from the other side of the conversation. While there are other modalities of input context $x$, for the purpose of this thesis, we will focus on the case where the input context is also a sequence of words $x = x_0 \ldots x_I$.

The standard approach to incorporate the input context to the sequence models is the *encoder-decoder* framework:

**Decoder**  The decoder is similar to the sequence model we defined in the previous section (without the sequence-level input context). For the recurrent model, the only difference is
that an extra input context vector $c_j$ is added to Equation 2.2 at each time step $j$:

$$h_j = f_{dec}^r(h_{j-1}, e_{j-1}, c_j) \quad (2.8)$$

The reason we need to build different context vector $c_j$ to represent $x$ at each time step $j$ is that the part of the input sequence we want to focus on when generating $y_j$ at a specific position $j$ is different. This leaves two questions: (1) how to encode the input context into a vector $c_j$, and (2) how to modify the model defined above to incorporate this new input vector? The first question will be discussed in other modules of the model. As for the second question, the way recurrent models incorporate input context is to concatenate the context vector $c_j$ with the word embedding input $e_{j-1}$ before passing through a linear transformation. This transformed vector is then used as the input to the recurrent unit as usual:

$$h_j = \sigma(W^h h_{j-1} + W^e[e_{j-1}; c_j]) \quad (2.9)$$

On the other hand, for a self-attention (Transformer) model, similarly, we also need to incorporate an extra input context vector $c_j$ to the model. The method here is even simpler:

$$h_j = \sum_{j'=0}^{j-1} a_{j',j} W^e e_{j'} + c_j \quad (2.10)$$
CHAPTER 2. BACKGROUND

Here, $c_j$ is required to have the same dimension as $h_j$.

We now look at how to encode the input context into the context vector $c_j$. This is generally done by two modules: (1) an encoder that encodes the embedding of the words in the input sequence $e_0^x, \ldots, e_I^x$ into a series of vectors $h_0^x, \ldots, h_I^x$, and (2) an encoder-decoder attention that composes the series of vectors $h_0^x, \ldots, h_I^x$ into a single context vector $c_j$.

**Encoder** The encoder, again, is very similar to the model we defined in Section 2.2.1. The defining differences, however, are different from the decoder and in two-folds. First of all, because the output of the module is not a probability distribution, but rather just simply a series of encoded vectors $h_0^x, \ldots, h_I^x$, we no longer need the output layer $f_o$ and the softmax function. Secondly, because we have access to the full input sequence $x = x_0 \ldots x_I$, at a certain position $i$ of the input sequence, we don’t only want to encode the information in its prefix, but also the suffix. For recurrent models, the solution is to have two models build encodings from left-to-right (forward) order and right-to-left (backward) order, and then concatenate them as the final context vectors. For self-attention models, the solution is simply to attend to both the embedding of the words that come before and after position $i$ (instead of just before, as in Equation 2.6).

**Encoder-Decoder Attention** Similar to the idea of “self-attention” as introduced in Section 2.2.1, we also compose the series of vectors $h_0^x, \ldots, h_I^x$ into a single context vector "$c_j"
CHAPTER 2. BACKGROUND

c\_j by performing a weighted sum of them.

\[ c\_j = \sum_{i=0}^{l-1} a\_ij W^q h^x_i \quad (2.11) \]

Now we need to compute the attention weights \( a\_ij \). Note again, that we may want to focus on different parts of the input sequence depending on specific target context at time step \( j \). One of the ways to compute the attention weights is build another attention module by reusing the attention mechanism in Equation 2.7, but change the query into some version of the decoder state. For example, for self-attention (Transformer) models, Vaswani et al. (2017) uses the intermediate state output (before the layer normalization and residual connections) as the query.

Similar to the definition of \( f_r \) for recurrent models, there are also a wide variety of attention weight computations that have been proposed in the literature. For example, the first paper that proposed the encoder-decoder attention mechanism (Bahdanau, Cho, and Bengio, 2014) proposed to use a feed-forward network (also called “concat” attention in some papers) instead of dot-products to compute attention weights. Luong, Pham, and Manning (2015a) is the first to propose using dot product in attentions, which further divides into alternatives such as dot attention between vectors of the same dimension and general attention that adds a transformation so that it can work for any pair of vectors. It also proposed to integrate multiple attention of different scopes, either global or local. Instead
of having soft attention weights, (Xu et al., 2015; Shankar, Garg, and Sarawagi, 2018; Wu, Shapiro, and Cotterell, 2018) also proposed to use hard 0-1 attention for the image caption generation, neural machine translation, transliteration and morphological inflection task, etc. Deng et al. (2018) further modeled the attention weights as a latent variable and used variational inference to make both training and inference tractable. Vaswani et al. (2017) extended the use of attention beyond encoder-decoder attention, where they created the query-key-value formulation, and also proposed to use multiple attention heads so the model can focus on multiple different positions of the sequence at the same time.

To summarize this section, Figure 2.1 is an illustration of the encoder-decoder architecture that we discussed. Note that for both the encoder and decoder, we drew a recurrent architecture here. For the encoder-decoder attention module, we only draw the computation graph for $c_J$ (while all the other context vectors are also computed by the encoder-decoder attention).

### 2.3 Explanation

#### 2.3.1 Problem Definition

As the reader can see from Section 2.2, each prediction $y_j$ made by the model corresponds to some input feature set $\mathcal{F}$. As stated in the previous section, for the discussion in this thesis,
Figure 2.1: A summary of the encoder-decoder architecture.
we want to find important input tokens in the feature set $\mathcal{F}$ for a particular prediction $y_j$. To formally define the concept of importance, we now define an importance distribution $\psi(f)$ over the input feature space $f \in \mathcal{F}$ that assigns an importance score to each input feature $f$ for this prediction $y_j$. And since $\psi(f)$ is a distribution, it follows that $\sum_{f \in \mathcal{F}} \psi(f) = 1$. Our explanation task now is essentially finding this importance distribution $\psi(f)$.

To exemplify, we define the task under two specific sequence generation tasks:

- For a language model, the input feature set $\mathcal{F}$ is defined as the tokens in the generated prefix $s_j = y_0y_1 \ldots y_{j-1}$. For a particular predicted token $y_j$, we are establishing an importance distribution over the tokens in its prefix.

- For a machine translation model, the input feature set $\mathcal{F}$ is defined as the tokens in the input sequence $x = x_0x_1 \ldots x_I$. For a particular predicted translation token $y_j$, we are establishing an importance distribution over the input tokens.

The readers may notice that the definition of the input feature set $\mathcal{F}$ does not always align with the actual input feature set into the model for the prediction. Specifically, for the machine translation model, we are not including the tokens in the prefix $s_j$, which also serves as inputs to the model for the prediction $y_j$. The reason for such definition in the context of machine translation is that it aligns better with the problem of word alignment, proposed as one of the main challenges for neural machine translation in Koehn and Knowles (2017a), and even earlier as one of the important components for statistical
CHAPTER 2. BACKGROUND

machine translation (Brown et al., 1993; Koehn, Och, and Marcu, 2003). Depending on the problem of interest, the alternative definition to align $\mathcal{F}$ with the model’s actual input feature set is also sometimes adopted. For example, Li et al. (2019) analyze the importance distribution on both the words in the source input and the target prefix.

It should be noted that while the studies on model explanation covered in this thesis are mostly concerned with feature importance, explanation as a category of auxiliary signals can actually take many different forms. Other than the global explanation methods we introduced in Section 2.1.2, there is another stream of work named “Example-based Explanations” in Molnar (2019) that focuses on selecting some specific data instances to explain the behavior of a machine learning model. Examples of such methods include identifying test examples that will change the model’s prediction\(^5\) and identifying training examples that is most important for a specific prediction (Khandelwal et al., 2020; Khandelwal et al., 2021, k-NN based explanation). Similar to importance on a token, importance on single neurons has also been defined and studied in some previous work (Bau et al., 2019; Dalvi et al., 2019a; Dalvi et al., 2019b). Apart from example-based approaches, there is also prior work on generating explanations in the form of natural language text (Marasović et al., 2020; Wiegrefe, Marasović, and Smith, 2021). This is by no means an exhaustive list, as building user-friendly explanations for AI decisions is a fast-evolving field where new tools

\(^5\)This is often referred to as “counterfactual explanations” (Kim, Koyejo, and Khanna, 2016) or “adversarial examples” (Goodfellow, Shlens, and Szegedy, 2015; Cheng et al., 2020a; Bojchevski and Günnemann, 2018, \textit{inter alia})
CHAPTER 2. BACKGROUND

are constantly being built (Strobelt et al., 2019; Wexler et al., 2020) and user experiences are constantly being evaluated (Bove et al., 2022). The variants of model explanations are further enriched by the fact that each form can have many implementations in terms of design choices, such as visualization (Bach, Huang, and Al-Onaizan, 2011; Kepler et al., 2019b; Shenoy et al., 2021) and interface (Bove et al., 2022).

While a rigorous comparison of user friendliness of model explanations is out of the scope of this thesis, we would like to argue that the problem context plays an important role in selecting the most appropriate form of explanations – specifically, the goal of the runtime model audit task and the focus on sequence modeling. Similar to the argument we made in Section 2.1.1, any explanation we generate should help users detect and fix anomaly output. For example, while example-based explanations and neuron-level analysis can highlight potential issues in the data and defects in the model design, they are not of primary interest for audits conducted at runtime. Similarly, while natural language explanations can be perceived helpful as elaborations for a classifier prediction, they are likely not the most efficient way to highlight problems in a model-generated sequence. These considerations drive us to focus on our previously defined form of model explanations.
CHAPTER 2. BACKGROUND

2.3.2 Methods of Explanations

Post-hoc Explanation Based on the Original Model

Gradient-based Methods/Saliency    We are going to introduce these methods in two steps. We start by talking about how the feature importance is defined on a single dimension of the word embedding (denoted as $\tilde{\psi}$), followed by an introduction of the composition schemes, an issue that specifically arises when these methods are applied to NLP.

The gradient-based explanation methods or saliency method (Simonyan, Vedaldi, and Zisserman, 2013) and its variants, are among the most adopted methods of model explanation.

To illustrate the intuition behind this method, let’s consider the probability of a specific output word $y_j$ when the $d$-th dimension of a input word embedding $e_{i,d}$ is perturbed from its original value $e_{i,d}^*$. Then, for a perturbed input $x_+$, constructed by the $d$-th dimension of the embedding being perturbed as $e_{i,d}^{*+} = e_{i,d}^* + \varepsilon$, the updated probability can be estimated using first-order Taylor expansion:

$$p(y_j \mid s_j, x_+) \approx p(y_j \mid s_j, x) + \left. \frac{\partial p(y_j \mid s_j, x)}{\partial e_{i,d}} \right|_{e_{i,d}^*} \cdot \varepsilon \quad (2.12)$$

What this formula tells us is that the sensitivity of the probability $p(y_j \mid s_j, x)$ to a small perturbation $\varepsilon$ is determined by the gradient of the probability $p(y_j \mid s_j, x)$ with regard to $e_{i,d}^*$. 

\footnote{Since the normalization of feature importance happens after composition, feature importance on a single dimension ($\tilde{\psi}$) defined here will not be normalized, unlike the final feature importance function $\psi$.}
CHAPTER 2. BACKGROUND

the $d$-th dimension of the word embedding $e_{i,d}$. Below we define $g(e_{i,d}^*) = \frac{\partial p(y_j|s_j,x)}{\partial e_{i,d}^*} |_{e_{i,d}^*}$ for conciseness. Intuitively, high sensitivity of the output score with regard to a certain feature indicates high importance of that feature and vice versa. Hence, we can define the feature importance as the aforementioned gradient value:

$$\tilde{\psi}(e_{i,d}^*) = g(e_{i,d}^*)$$  \hspace{1cm} (2.13)

This is called Saliency (Simonyan, Vedaldi, and Zisserman, 2013). Straight-forward as it is, recall that the remainder term of the first-order Taylor expansion is determined both by the second-order derivative of $p(y_j|s_j,x)$ and $\varepsilon^2$. Hence, when the curvature of $p(y_j|e_{i,d}^*)$ is large, such function approximation will only work within a very small neighborhood, thus causing the sensitivity estimation to be noisy. To alleviate this issue, an intuitive method is to sample multiple points around $e_{i,d}^*$ to compute gradients and average them, thus artificially expands the validity neighborhood of the estimation.

$$\tilde{\psi}(e_{i,d}^*) = \frac{1}{K} \sum_{k=1}^{K} g(e_{i,d}^* + \varepsilon_k)$$  \hspace{1cm} (2.14)

where $\varepsilon_k \sim \mathcal{N}(0, \sigma^2)$. This is proposed as SmoothGrad in Smilkov et al. (2017).

The last gradient-based method we are going to introduce is Integrated Gradients (Sundararajan, Taly, and Yan, 2017). Motivated by a line integral formulation from the
“baseline” to the actual input from the data, they propose a different strategy to sample the points to evaluate the gradients:

\[
\tilde{\psi}(e_{i,d}^*) = \frac{1}{K} \sum_{k=1}^{K} g(\hat{e} + \frac{k}{K} \cdot (e_{i,d}^* - \hat{e}))
\]  

(2.15)

where \(\hat{e}\) is a baseline point defined as “where the prediction is neutral”. For a word embedding in NLP, we typically use \(\hat{e} = 0\).

We have so far defined several different versions of importance function \(\tilde{\psi}\) for a single dimension of the word embedding. However, the explanation problem that we set out to solve is to identify the most important input token for a specific prediction, which requires us to assign a scalar importance score for each token. Hence, a composition scheme is necessary to compose a vector element-wise importance \(\tilde{\psi}\) into a scalar token-level importance \(\psi\). An example of such composition scheme is to use the 1-norm of the element-wise importance vector (Li et al., 2016):

\[
\psi(e) = \frac{\left\| \tilde{\psi}(e) \right\|_1}{\sum_{e' \in \mathcal{F}} \left\| \tilde{\psi}(e') \right\|_1}
\]  

(2.16)

We will point out some problems both theoretically and empirically in Chapter 3, and use these insights to motivate another composition scheme.

**Other Back-Propagation-based Methods**

Beyond saliency methods, there are other
model explanation methods that are back-propagation-based. The key difference between those other methods and saliency is that, rather than following the propagation rules of the gradient computations, they define their own rules to quantify the influence to the final prediction from a unit in layer $l+1$, and recursively propagate back to the units connected to it at layer $l$. We survey two different methods here: $\varepsilon$-LRP (Bach et al., 2015) and DeepLIFT (Shrikumar, Greenside, and Kundaje, 2017).

- **$\varepsilon$-LRP**  Layer-wise Relevance Propagation (Bach et al., 2015, LRP) operates under the “conservation property”, that all of the activations has been received by a neuron must be redistributed to the lower layer. The $\varepsilon$-LRP rule is among the simplest rules. Suppose we are looking at the neural network layer $l$ and $l+1$. In the forward pass, a neuron activation $a_k$ in layer $l+1$ is computed from the $m$ activations in layer $l$ as follows:

$$a_k = \max(0, \sum_{j=0}^{m} a_j w_{jk})$$  \hspace{1cm} (2.17)

where $w_{jk}$ is the corresponding weights of the edge that connects neuron $j$ and $k$. Now we turn to the back-propagation rule from layer $l + 1$ to layer $l$. The relevance
CHAPTER 2. BACKGROUND

\( R_j \) to a neuron in a layer is defined by the \( \varepsilon \)-LRP rule as follows:

\[
R_j = \sum_{k=0}^{n} a_j w_{jk} \varepsilon + \sum_{m,j' = 0}^{\infty} a_{j'} w_{j'k} \varepsilon R_k
\]

(2.18)

where \( n \) is the number of the neurons in \( l+1 \) that the neuron \( j \) is connected to. \( \varepsilon \) is a small positive number that avoids the divide-by-0 error.

We refer the readers to Montavon et al. (2019) for other variants of the LRP rules.

- **DeepLIFT** The idea behind DeepLIFT is very similar to \( \varepsilon \)-LRP rules, except that it uses a baseline during the process. Instead of attributing the activation values themselves, it attributes the change of activation values when the input is changed from the baseline input to the actual input from the data. Correspondingly, the DeepLIFT backpropagation rules will be defined as:

\[
R_j = \sum_{k=0}^{n} a_j w_{jk} - \hat{a}_j w_{jk} \sum_{m,j' = 0}^{\infty} a_{j'} w_{j'k} - \hat{a}_{j'} w_{j'k} R_k
\]

(2.19)

where the new \( \hat{a}_j \) introduced here is also the activation level for neuron \( j \), but when the baseline input is used.

Since all of the methods above require modifying the back-propagation rules for gradients, it is hard to come up with an architecture-agnostic way of implementing these methods
by leveraging the auto-differentiation module. Normally, expensive re-implementations of custom modules with alternative back-propagation rules need to be performed, which is how LRP was applied for the analysis of neural machine translation models in Ding et al. (2017), Voita et al. (2019), and Voita, Sennrich, and Titov (2021). Ancona et al. (2018) proposed an alternative approach that enables easier re-implementation of some method variants with a unified back-propagation rule that simply adds some extra operations on top of the usual back-propagation process. Besides, there were also some separate reports about $\varepsilon$-LRP not being able to adapt to other non-linearities beyond ReLU (Ancona et al., 2018), or to skip connections and attention operators (Chefer, Gur, and Wolf, 2021). Ancona et al. (2018) further reported similar concerns for DeepLIFT on RNN and LSTM units. Because of these concerns, we will not be studying those methods within this thesis.

**Perturbation-based Methods** Another category of methods to generate model explanations is by perturbing some of the input features. The motivation is similar to back-propagation-based methods: if a certain input feature is important, the model output score will be sensitive to perturbations on such feature. Hence, other than measuring the importance of a feature by gradients, we can also simply measure it by introducing perturbations to its original input value and observing the change in the output score. Note that this is slightly different from adversarial sample generation, which aims to reveal the space of pathological inputs that would lead to prediction errors.
CHAPTER 2. BACKGROUND

In computer vision, Zeiler and Fergus (2014) first proposed the method of occlusion to cover up an image block in order to analyze the importance of the input pixels. Zintgraf et al. (2017) further extended this approach by introducing a conditional sampling procedure. On the other hand, in the NLP context, Li, Monroe, and Jurafsky (2016) proposed representation erasure, where either individual dimensions or the whole vector of the word embedding is set 0 in order to measure the difference in the prediction score. Li et al. (2019) adopted a similar method to Li, Monroe, and Jurafsky (2016) to neural machine translation by deterministically mask out each individual word in the input sentence.

Another perturbation-based model explanation method is Shapley value (Shapley, 1953; Strumbelj and Kononenko, 2014), which, in its original form, enumerates all possible feature values to compute feature importance under a game-theoretic framework. This is obviously very expensive and is not even feasible when the feature space is in $\mathbb{R}^n$, as with the case of neural networks. Hence, sampling-based methods are proposed by taking advantage of the connection between the Shapley value definition with DeepLIFT and Integrated Gradients (Chen, Lundberg, and Lee, 2019).

Post-hoc Explanation Based on a Dedicated Model

We mentioned in Section 2.3.2 that Saliency is essentially building a first-order Taylor approximation of the model’s behavior in the neighborhood of the original model input. In
other words, this is a linear surrogate model $M'$ built to approximate the behavior of the original model $M$ under a strict condition, with the only parameter of the model deterministically estimated by the gradient. Following this path, we can see another alternative path to explain the model prediction: we can build an intrinsically explainable surrogate model to approximate the original model’s behavior, and then use this surrogate model to explain the prediction of the model.

One popular approach that follows this path is Local Interpretable Model-Agnostic Explanations (Ribeiro, Singh, and Guestrin, 2016, LIME). The idea is exactly building a local interpretable surrogate model $M'$ (such as logistic regression with LASSO regularizer) to approximate the behavior of the original model $M$ in the neighborhood of a certain input-output pair $(x, y)$. Note that, to conform with our problem definition, the surrogate model $M'$ should have certain structures that can be interpreted as the importance of a specific input feature. For example, a logistic regression model conforms with our definition because the feature weights can be interpreted as this kind of importance measure.

The key difference between LIME to Saliency is that LIME does not deterministically estimate the parameters. Instead, it constructs a small training set $\mathcal{X}_L$ by introducing noises to the original input $x$, and feeds $\mathcal{X}_L$ into the original model $M$ to obtain corresponding outputs $\mathcal{Y}_L$. The small training set $\{\mathcal{X}_L, \mathcal{Y}_L\}$ is then used to estimate the parameters of the surrogate model $M'$ with the parameter estimator of choice (the most common example is
maximum likelihood estimation).

Although not commonly presented as a model explanation method in the literature, it is very common in the machine translation literature to use a word alignment model such as GIZA++ (Och and Ney, 2003) and FastAlign (Dyer, Chahuneau, and Smith, 2013a) to generate explanations for the predictions of sequence generation models (Tang, Sennrich, and Nivre, 2018; Stanovsky, Smith, and Zettlemoyer, 2019; Zhou, Gu, and Neubig, 2020, *inter alia*). On a very high level, the original idea of such models is to align word pairs of translation equivalence in an unsupervised fashion, by modeling the co-occurrence patterns of words in the sentence pairs that are translations of each other. When used for model explanations, the parameters of the word alignment model (as a surrogate model $M'$) is estimated on a parallel dataset either translated by humans or by the original model $M$. More details of the alignment models can be found in Koehn (2010).

**Intrinsic Explanation**

The final category of methods we would like to introduce is the set of models that are intrinsically interpretable. A part of this is essentially the same set of models that were used to build surrogate models $M'$ in Section 2.3.2. On top of that, people also frequently propose to incorporate an interpretable module in a black-box model intended to model a specific behavior. One example is the attention mechanism (Bahdanau, Cho, and Bengio, 2014)
and its variants (Luong, Pham, and Manning, 2015a; Xu et al., 2015; Shankar, Garg, and Sarawagi, 2018; Wu, Shapiro, and Cotterell, 2018; Deng et al., 2018; Vaswani et al., 2017) in sequence generation, which we have visited in Section 2.2.2. For sequence generation tasks with more copying behaviors, extra modules such as pointer networks (Vinyals, Fortunato, and Jaitly, 2015; See, Liu, and Manning, 2017) and copy mechanism (Gu et al., 2016) were also introduced. Apart from designing interpretable modules, there are also efforts to interpret certain weights of an existing module for model explanations. For example, cell decomposition (Murdoch and Szlam, 2017) and contextual decomposition (Murdoch, Liu, and Yu, 2018; Jumelet, Zuidema, and Hupkes, 2019) proposed to look at the activations of context gates in LSTM in order to study the word importance as well as the interactions between the context gates.

As we previously mentioned in Section 2.1.2, despite its simplicity due to dropping the surrogate model, there are some downsides to this approach. In terms of the concern over performance, a significant example in the context of sequence generation is machine translation, which saw a paradigm shift from the more interpretable statistical machine translation framework to the less interpretable neural machine translation framework in recent years (Koehn and Knowles, 2017b). Some of the variants of attention mechanism also face efficiency issues due to the expensive but required sampling procedure involved to train these modules. As for the concern over the reliability of the inherently interpretable
modules, one such example in the scope of model explanation is again regarding the attention mechanism, which had been deemed as interpretable upon its proposal, but recent studies showed that it might be prone to variations and adversarial manipulations (Serrano and Smith, 2019; Jain and Wallace, 2019; Wiegreffe and Pinter, 2019; Grimsley, Mayfield, and R.S. Bursten, 2020).

2.3.3 Evaluation of Explanations

Explanations for model predictions help users understand the underlying decision mechanisms of machine learning models, but they could also potentially be misleading. As we just mentioned, several studies found the explanation generated by the attention mechanism to be subject to variations and adversarial manipulations. In the context of computer vision, Alvarez-Melis and Jaakkola (2018) and Yeh et al. (2019b) both point out that the relative change rate of feature importance for some image classification tasks might be too large. Ghorbani, Abid, and Zou (2019) reveals the fragility of the model explanations by proposing two systematic gradient-based attacks, which produce adversarial examples that have high visual similarity but very different feature importance attributions. Similarly, Kindermans et al. (2019) found that adding a simple constant shift to the input image could cause incorrect feature importance attributions. All those studies support the necessity for evaluating the quality of model explanations before adopting them for model audit purposes.
CHAPTER 2. BACKGROUND

However, the evaluation of explanations is still an open research question, and there is no unified metric to perform such evaluations. This is largely due to the fact that explanation is largely a subjective matter – even two different humans can give very different explanations for the same decision. As a result, it is far from straightforward to formally define the desiderata of a “good” or “correct” explanation. Because of the difficulty in obtaining a well-defined evaluation metric, many studies build their conclusion upon the assumption that the explanation is always correct or only perform small qualitative sanity checks. Yet, most of the definitions concentrate around two criteria: (1) agreement with existing human intuitions (plausibility); (2) correctly and robustly reflect importance attribution (faithfulness).\footnote{The two criteria is taken from Jacovi and Goldberg (2020) and will be revisited in Chapter 3.}

Plausibility

For image classification tasks, the plausibility evaluation is pretty straightforward because human-labeled ground-truth bounding boxes for the object is often provided with the datasets, such as PASCAL VOC dataset (Everingham et al., n.d.), MSRA dataset (Liu et al., 2007), ImageNet (Deng et al., 2009), ILSVRC challenge dataset (Russakovsky et al., 2015). With these bounding boxes, common plausibility evaluation task includes the following (both coming from (Zhang et al., 2016)):

- **Pointing game**: compute the ratio of examples where the top-1 highest importance
CHAPTER 2. BACKGROUND

pixel falls into the ground-truth bounding box;

- **Object Localization**: generate a bounding box with the feature importance attribution and compute intersection-over-union (IOU) error with the ground-truth bounding box.

On the other hand, the landscape with NLP is much less clear, since the availability of ground-truth explanation is highly dependent on the task. For example, for the sentiment analysis task, word-level annotations of sentiment are available through the Stanford Sentiment Treebank (Socher et al., 2013, SST-2), and plausibility evaluations can directly be conducted through consistency evaluation with the ground-truth (Li, Monroe, and Jurafsky, 2016; Kim et al., 2020; Tsang, Rambhatla, and Liu, 2020). Some ground-truth annotations that are specifically targeted for the evaluation of model explanations were also created (DeYoung et al., 2020; Sen et al., 2020). For other tasks where ground-truth is not available, either the scope of the evaluation is constrained in order to take advantage of the existing human annotation (Poerner, Schütze, and Roth, 2018; Ding and Koehn, 2021), or toy task is designed to obtain trivial ground-truth for evaluation (Arras et al., 2019).

It should be noted that there were also a few studies that skipped human annotations altogether and directly designed interactive evaluations that involve human evaluators of the explanations. For example, Narayanan et al. (2018) conducted human evaluations for various cognitive properties on two hand-crafted, synthetic decision-making task.
CHAPTER 2. BACKGROUND

Graber (2019) evaluated elastic-search-based interpretation for question answering models by measuring whether these interpretations improve human performance in Quizbowl. Nguyen (2018) asked humans to simulate the model behavior based on the model explanation and compute the accuracy against the actual model predictions. Hase and Bansal (2020) proposed to ask humans to simulate the model behavior with and without the explanation and use the differences between the two simulations to quantify the quality of model explanations.

**Faithfulness**

There are two different ways to evaluate faithfulness in the existing literature.

**Perturbation** This is often used to falsify a model explanation method. The idea is to introduce perturbations to the input that should not affect the model’s decision mechanism, and then measure the consistency of the explanation before/after the perturbations. Earlier in this section, we have already mentioned several studies in computer vision that used such technique to point out deficiencies of model explanations (Alvarez-Melis and Jaakkola, 2018; Yeh et al., 2019b; Ghorbani, Abid, and Zou, 2019; Kindermans et al., 2019). In NLP, this line of study is largely missing because of the difficulty in automatically generating perturbed discrete examples while preserving the original model decision mechanism.

**Incremental Insertion/Deletion** As we repeatedly mentioned during the introduction of model explanation methods, the model’s prediction score should be most sensitive to
the changes in the most important feature. Hence, we can incrementally insert or delete the most important remaining feature to the model input and analyze the change in the model’s prediction score of the same class. If the importance attribution is close to optimal, a relatively significant increase/drop should occur earlier in the insertion/deletion sequence, as the most important feature should be inserted/deleted first. Such effect can be quantified by the area-under-curve (AUC) of the model prediction score. For incremental insertion and deletion, we deem a larger and smaller AUC to be better, respectively. This was proposed in (Samek et al., 2017) and was used in Shrikumar, Greenside, and Kundaje (2017), Nguyen (2018), and Petsiuk, Das, and Saenko (2018), *inter alia*. A similar idea was also proposed in Arras et al. (2016), but assumes the existence of a ground-truth class.

A criticism on this method from (Ju et al., 2021) pointed out that this evaluation is essentially doing the same thing as an explanation method called occlusion (Zeiler and Fergus, 2014), which we have covered in the previous section. Hence, this evaluation falls into the logic trap of evaluating model explanations with an alternative explanation, which may also be prone to errors.

Recent study (Hooker et al., 2019) proposed an update to this method, where new models are trained and evaluated each time a proportion of the most important features are removed. Subsequently, Li et al. (2020) adopted this method in the context of neural machine translation.
CHAPTER 2. BACKGROUND

2.4 Quality

2.4.1 Problem Definition

As users of a machine learning model, it is often useful to get a sense of the quality or the confidence of the prediction along with the prediction itself. Having such information allows us to calibrate our expectations for the model and interfere when an error is imminent. There are many levels such estimation of output quality could happen: at system-level, sequence-level, or word-level. In this thesis, we will focus on word-level quality estimation, which offers the most fine-grained quality signals to the users.

Formally, the problem is defined as follows: given a word sequence \( y = y_0 y_1 \ldots y_J \) generated by a sequence model, we would like to generate:

- binary word quality labels \( l^w = l^w_0 l^w_1 \ldots l^w_J \), with \( l^w_j \in \{ \text{OK}, \text{BAD} \} \) corresponding to the quality of \( y_j \)

- (optional) binary gap quality labels \( l^g = l^g_0 l^g_1 \ldots l^g_{J-1} \), with \( l^g_j \in \{ \text{OK}, \text{BAD} \} \) corresponding to the quality of the gap between \( y_j \) and \( y_{j+1} \)

The purpose of having gap quality labels is to capture potential missing words that should be inserted in the generated sequence.

One slight complication occurs when subword segmentation was used to reduce the
vocabulary size. In that case, the token that is generated at each time step will be a subword instead of a word, and the exact segmentation will depend on the segmentation dictionary and algorithm that is used to perform the segmentation. To simplify the problem, we define the symbols we perform the evaluation for to be full words. We will discuss algorithms that are used to perform the conversion of labels to and from subwords in Chapter 6.

### 2.4.2 Methods of Quality Estimation

We will follow the same general methodology of runtime audit introduced in Section 2.1.2. There is one key difference that distinct quality estimation from explanation: we will always need to wait until the prediction is made before making estimations of the quality. In other words, the methods will always be post-hoc, and the notion of intrinsic audit is no longer relevant here. Hence, in this section, we will review the methods following the two categories of post-hoc audit methods: original and dedicated.

It should be noted that while we will be using the name Quality Estimation for the category of methods discussed in this section (as a slightly broader concept compared to Confidence Estimation), this name is in fact first proposed by Specia et al. (2013) in the context of machine translation. In a lot of the papers we survey here, the same concept might be referred to as Confidence Estimation, Uncertainty Estimation, or Calibration, and may have slightly different task definitions depending on their names. For completeness,
we will cover all of these methods in this section, but the readers should be aware of the subtle differences in their definitions and respective contexts as we go through them.

**Quality Estimation Based on the Original Model**

When relying on the original prediction model to make estimations of the quality, the estimation will primarily be based on the posterior probability from the prediction model.

**Calibration**  Calibration (Cosmides and Tooby, 1996) is a property of model probability aligning with the actual chance of getting a correct prediction. For example, if there are 10 model predictions where the model assigned a 0.9 probability, a *perfectly calibrated* model will get 9 of them correct. If less than 10 of them are correct, we say the model is *over-confident*, and if all of them are correct, we say the model is *under-confident*.

Calibration is a well-studied problem both for general machine learning and for NLP. Guo et al. (2017) is the first work that studied the property of calibration for neural networks, and largely laid the foundation for subsequent studies of this problem by setting up the evaluation framework and adopting Expected Calibration Error (Naeini, Cooper, and Hauskrecht, 2015, ECE) as the evaluation metric. The analysis revealed that factors like model capacity, normalization, and regularization all have strong effects on model calibration. It further proposes a re-calibration technique (similar to that of Platt (2000), Zadrozny and Elkan (2001), and Zadrozny and Elkan (2002), although not for neural networks) of...
CHAPTER 2. BACKGROUND

temperature scaling to adjust the output probability in order to improve model calibration. On top of this re-calibration technique, Pereyra et al. (2017) and Kumar, Sarawagi, and Jain (2018) proposed alternative loss functions during training that improve model calibration. Vaicenavicius et al. (2019) extended the evaluation framework in Guo et al. (2017) and developed a more general mathematical framework that can better evaluate the calibration of multi-class classifiers.

In the context of language modeling, Braverman et al. (2020) observed several state-of-the-art language models to be miscalibrated and proposed an algorithm called entropy rate calibration to mitigate such effect. Kong et al. (2020) on the other hand, proposed to mitigate such effect by performing regularized finetuning. Jiang et al. (2021) made similar miscalibration observations regarding the task of LM-based question answering (QA), and studied the effect of several calibration improvement techniques including temperature scaling and finetuning. For neural machine translation, Kumar and Sarawagi (2019) and Wang et al. (2020c) studied the calibration for the probability from neural machine translation models. While the probability is well-calibrated during training (Kumar and Sarawagi, 2019), Wang et al. (2020c) observed a significant over-confidence phenomenon during inference and introduced a novel graduated label smoothing loss function to improve the calibration.

Intuitively, having a well-calibrated model is helpful for communicating the model’s
confidence to the users or to the subsequent steps in the inference pipeline, but we will show in Chapter 5 with a toy example that it takes more than good calibration to accomplish our defined problem.

**Bayesian Uncertainty Estimation** For studies in model calibration, we are still only obtaining a single point estimation for the probability \( p(y_j^* \mid s_j, x; \theta^*) \), with \( y_j^* \) being the argmax prediction from distribution \( p(y_j \mid s_j, x; \theta^*) \) and \( \theta^* \) being the parameter of the model. From a Bayesian perspective, this is a very rough estimation because only one sample \( \theta^* \) is drawn from the parameter distribution \( p(\theta) \). Ideally, instead of conducting only one single point estimate, the parameters of the model should be marginalized out during inference:

\[
p(y_j^* \mid s_j, x) = \int p(y_j^* \mid s_j, x; \theta)p(\theta)d\theta \tag{2.20}
\]

The posterior estimation of \( p(y_j^* \mid s_j, x) \) is then used as the uncertainty metric.\(^8\) However, a closed-form solution to Equation 2.20 is apparently not tractable.

Various solutions have proposed different approximations to this. Gal and Ghahramani (2016) first proposed *Monte Carlo Dropout*, which approximates Equation 2.20 by performing dropout on the neurons of each layers. The dropout process is repeated \( k \) times, thus

\(^8\)In the literature, the variance of the estimate with different \( \theta \) and the entropy of the posterior distribution \( p(y_j^* \mid s_j, x) \) have also been used.
equivalent to creating \( k \) samples of \( \theta \). The approximation is then made by:

\[
p(y_j^* | s_j, x) = \frac{1}{K} \sum_{k=0}^{K} p(y_j^* | s_j, x; \theta_k)
\]  

(2.21)

McClure and Kriegeskorte (2016) extended the study in Gal and Ghahramani (2016) by exploring other sampling options such as dropping connections and adding Gaussian noise. Brach, Sick, and Dürr (2020) proposed a single-shot Monte Carlo Dropout variant by considering the value statistics of neurons on the same layer. Tsymbalov, Fedyanin, and Panov (2020) proposed to improve Monte Carlo Dropout by encouraging the samples to come from a diverse set of neurons in order to achieve faster convergence, which is achieved by sampling the dropout masks from a Determinantal Point Process (Macchi, 1975; Kulesza and Taskar, 2012, DPP).

Monte Carlo Dropout and its variants have also been widely adopted in natural language processing. Dong, Quirk, and Lapata (2018) first adopted Monte Carlo Dropout to estimate uncertainties of the predictions from semantic parsers. Xiao and Wang (2019) conducted a similar study for various Bi-LSTM text classification tasks such as sentiment analysis and named entity recognition (NER). Shelmanov et al. (2021) extended the above study to classification models finetuned from pre-trained Transformers. They also studied the DPP variant of Monte Carlo Dropout (Tsymbalov, Fedyanin, and Panov, 2020) and obtained mixed results when comparing to the basic version. Xiao, Gomez, and Gal (2020) proposed
several sequence-level alternative uncertainty measures to Monte Carlo Dropout, including beam score, sequence-level log probability, and BLEU Variance under random dropout.

**Application in Machine Translation**  Most related to our study, in the neural machine translation context, Ott et al. (2018) analyzed the probability output from neural machine translation models and its interaction with beam search. They found beam search to work well but the model generally has high uncertainties with its predictions. Wang et al. (2019d) proposed alternative uncertainty measures based on Monte Carlo Dropout and used them to reweigh training data generated from back-translation through modified loss functions. Fomicheva et al. (2020) adopted the uncertainty measures proposed in the previous paper for sentence-level quality estimation of machine translation outputs and obtained competitive results when compared to the widely-adopted Predictor-Estimator approach (Kim, Lee, and Na, 2017; Kepler et al., 2019b).

**Quality Estimation Based on a Dedicated Model**

We now turn to the alternative approach that builds the estimation of quality based on a dedicated confidence model. A lot of the studies in this category still carry the uncertainty estimation goal, but we will also visit some of the most relevant studies that motivated our problem definition and evaluation framework.

**Bayesian Uncertainty Estimation**  The simplest such approach is to build a deep ensemble
CHAPTER 2. BACKGROUND

(Lakshminarayanan, Pritzel, and Blundell, 2017), which is just training \( k \) copies of the same model with random initialization on a random sample of the training data. The \( k \) models will be used to build an ensemble as a uniformly-weighted mixture model. Optionally, the training data can also include adversarial samples generated with the previous model, in a similar fashion as Boosting with building random forests (Friedman, 2001). Zhang, Kailkhura, and Han (2020) proposed a similar approach but allowed tuning the ensemble mixture weights.

**Meta-Model** A slightly more parameterized approach is to build a dedicated “meta-model” or “confidence model” based on the original prediction model. For example, Corbière et al. (2019) added an extra regression head (called “ConfidNet”) sharing the same input as the classification head that makes the model prediction, which is trained to reproduce confidence metrics computed from the prediction score of the classification head. The prediction model is trained before this regression head is added, and all the parameters of the prediction model are frozen during the regression head training to avoid degradation of the prediction performance. Chen et al. (2019) built simple logistic regression “probes” with the representation built at each level of a deep neural network as input features and use the probes’ predictions to assess the certainty of the model prediction.

There were also studies that proposed alternative loss functions that jointly train the model to learn to predict and estimate confidence. One such example is Rejection Option
(Cortes, DeSalvo, and Mohri, 2016), which jointly learns a prediction head \( p(y \mid x) \) and a rejection head \( r(y \mid x) \). While the prediction head makes predictions, as usual, the rejection head outputs the probability of “rejecting” this example – that is, to abstain from making a prediction for this example. The loss function of Rejection Option is shown as follows:

\[
L(x, y) = - \sum_{k=1}^{K} (\log p(y \mid x) 1_{r(y|x) < \frac{1}{2}} - c 1_{r(y|x) \geq \frac{1}{2}})
\]

(2.22)

where \( c \) is a hyperparameter that controls the fraction of rejected samples by setting a soft threshold for the log probability. However, it is not intuitive regarding what \( c \) value should take if one wishes to reject a certain fraction of samples. Geifman and El-Yaniv (2019) proposed an improved risk-based formulation to resolve this issue. In this formulation, we are essentially trying to balance two different sources of risks:

\[
\text{Coverage Risk: } \phi(r) = \mathbb{E}_{x,y \sim D} r(y \mid x)
\]

(2.23)

\[
\text{Selective Risk: } R(p, r) = \frac{-\mathbb{E}_{x,y \sim D}(\log p(y \mid x)(1 - r(y \mid x)))}{\phi(r)}
\]

(2.24)

In this formulation, the selective risk aims to only select the predictions with the highest confidence while the coverage risk enforces an adequately broad coverage. The optimization problem is then reformulated as minimizing \( R(p, r) \) while subject to the constraint that \( \phi(r) < 1 - c \), with \( c \) now being the target example coverage of the model. Other than the
CHAPTER 2. BACKGROUND

Rejection Option, another example of modifying the loss functions is Moon et al. (2020), where a correctness ranking loss term is added to help the model learn about prediction uncertainty.

**Similarity-Based Uncertainty Estimation** A final category of methods we would like to cover is estimating uncertainty by measuring the maximum similarity of the text example input to any examples in the training dataset. The intuition is that, if a test example input is really similar to something the model has seen in the training set, the model should be able to predict the output with high confidence. On the other hand, if there is nothing in the training set similar to the text input, or there are multiple training examples with different labeled classes and are all equally similar to the test example input, the predict confidence will be low. Jiang et al. (2018) captures such intuition by defining a trust score based on class density within the intermediate representation space of a neural network model. The score is computed at inference time by making k-nearest-neighbor (kNN) queries into a database of intermediate representations collected during training time on the training dataset. In the context of neural machine translation, Niehues and Pham (2019) estimated output uncertainty by adopting a similar kNN approach above, and also attempted to get rid of the expensive representation database by estimating the distance with the reconstruction loss from an autoencoder.

**Quality Estimation for Machine Translation** So far, none of the papers we have reviewed
has exactly the same problem definition as ours. We will now look at the line of study that directly motivated our problem definition and our evaluation framework, which is the task of quality estimation in the context of machine translation.

In the machine translation context, quality estimation (QE) is the task of estimating the quality of translation without access to a human-generated reference. As we have mentioned above, there are many levels at which such estimation could happen. In fact, a lot of recent advances on quality estimation (Rei et al., 2020; Thompson and Post, 2020a; Ranasinghe, Orasan, and Mitkov, 2020, *inter alia*) focus on estimating the quality of translation on either the corpus or segment-level. However, in practice, the end-users of machine translation (MT) often call for quality signals on a more fine-grained level—the level of individual words in a translation. Such signals are not only useful for more fine-grained audit of translation quality but also open up the potential for targeted post-processing and faster human post-editing. For the purpose of this thesis, we will focus on *word-level* quality estimation for this part of the literature review.

The earliest study that adopts a similar problem definition in machine translation dates back to Ueffing, Macherey, and Ney (2003), well before the inception of neural machine translation, around the same time as the phrase-based statistical machine translation. Together with several other contemporary studies (Blatz et al., 2004; Ueffing and Ney, 2005; Ueffing and Ney, 2007), they experimented with several different confidence models of a
target word $y_j$: (1) estimate the probability of a certain output word over a word graph using the forward-backward algorithm; (2) collecting word occurrence statistics over the N-best list using Levenshtein alignment; (3) leveraging a phrase model and a language model to score of all phrase pairs that contains $y_j$; (4) max IBM Model 1 alignment model score of $y_j$ over all the source words. Among these methods, Ueffing and Ney (2007) concluded that approach (3) (called “direct phrase-based confidence measure”) performed very well and that the performance of the confidence measures depends heavily on the definition of the reference tags.

As we have mentioned above, Specia et al. (2013) first proposed to use the term Quality Estimation instead of Confidence Estimation for this research area. Around the similar time, Bach, Huang, and Al-Onaizan (2011) and Luong (2012) proposed to use feature-rich classifiers to approach the problem, a methodology also adopted by Specia et al. (2013). Bojar et al. (2013) established the WMT shared task of word-level quality estimation which become the de-facto standard for evaluating methods of machine translation quality estimation ever since.

Upon the rise of deep learning, neural network classifiers were adopted for word-level quality estimation. The first work that adopted this approach (Kreutzer, Schamoni, and Riezler, 2015) used a feed-forward neural network classifier with a mixture of word embeddings and features as input. Martins et al. (2016) proposed RNN and Convolutional
CHAPTER 2. BACKGROUND

Neural Network (CNN) architectures for this task, and combined predictions from these architectures with a feature-rich feed-forward classifier to form an ensemble. Martins et al. (2017) and Martins, Kepler, and Monteiro (2017) further proposed to train the model on augmentation data generated by an automatic post-editing (APE) system.

The most recent paradigm shift came upon the invention of Transformer (Vaswani et al., 2017) and pre-trained models (Peters et al., 2018; Devlin et al., 2019, inter alia). This paradigm shift also largely eliminated the need for hand-crafted features for word-level quality estimation, an effort that started from the RNN-based proposal in Kim and Lee (2016). Well before the proposal of BERT (Devlin et al., 2019) and XLM (Conneau and Lample, 2019), Kim et al. (2017) proposed a very similar architecture to what is known today as Translation Language Model (Conneau and Lample, 2019, TLM). The paper also proposed a two-step training paradigm, known as Predictor-Estimator (PredEst), which is very similar to the current “pre-train and finetune” paradigm. The paradigm starts by training a TLM on a large-scale parallel corpus and then finetuning the model on a small corpus of translation triplets created by humans. The winning system of WMT 2018 from Alibaba (Wang et al., 2018) was the first to adopt a Transformer architecture variant (Fan et al., 2019) similar to the Predictor-Estimator architecture, but did not fully eliminate the use of the hand-crafted features. The next year, the Unbabel submission Kepler et al. (2019a) build upon these efforts and adopted the then-newly proposed BERT and XLM model to
CHAPTER 2. BACKGROUND

replace the predictor training step. The open-source codebase built from this submission (Kepler et al., 2019b), named OpenKiwi\(^9\), has been largely accepted as the state-of-the-art of word-level quality estimation since then.

Some recent advancements include building unsupervised models without human-labeled data (Fomicheva et al., 2020; Tuan et al., 2021), and formulating the problem of word-level quality estimation as finding rationales for sentence-level quality estimation systems (Treviso et al., 2021). But whether those methods can drive the next paradigm shift is an open question that awaits further exploration.

2.4.3 Evaluation of Quality Estimation

Like model explanations, there is no clear consensus as to how to evaluate methods of quality estimation. Below we introduce several most widely adopted methods of evaluating the quality estimation methods in the current literature.

**Expected Calibration Error (ECE)** ECE (Naeini, Cooper, and Hauskrecht, 2015) is a common method to evaluate model calibration, which we have covered in Section 2.4.2. The key idea is to group the total \(N\) model predictions into \(K\) bins \(B_0, \ldots, B_{K-1}\) by the

\(^9\)https://github.com/Unbabel/OpenKiwi
probability of the prediction. The metric is then computed by the following formula:

$$ECE = \frac{1}{K-1} \sum_{k=0}^{K-1} \left| \frac{|B_k|}{N} \right| \left| \text{acc}(B_k) - \text{conf}(B_k) \right|$$  \hspace{1cm} (2.25)

where \( \text{acc}(B_k) \) stands for the accuracy of the predictions in this bin, and \( \text{conf}(B_k) \) stands for the average prediction probability (confidence) in this bin.

Note that the above formulation is specifically proposed for classification problems with 2 classes. Guo et al. (2017) proposed an extension that enables application of this metric to multiple classes, which we will not cover here.

Nixon et al. (2019) has pointed out that ECE has several flaws associated with its binning mechanism, and has proposed solutions to counteract such flaws.

**Brier Score**  First proposed by Brier et al. (1950) and adopted by studies including Lakshminarayanan, Pritzel, and Blundell (2017) and Snoek et al. (2019), Brier Score can be perceived as the mean squared error adapted for classification problems. For a data instance with ground truth label \( y^* \), the Brier score is computed as:

$$BS = \frac{1}{|C|} \sum_{c \in C} (\mathbb{1}_{c=y^*} - p(y = c \mid x))^2$$  \hspace{1cm} (2.26)

where \( C \) is the set of output classes of the classifier, \( p(y = c \mid x) \) is the model prediction score for a certain class \( c \).
CHAPTER 2. BACKGROUND

**Area Under Curve (AUC)**  This is used to evaluate the quality of binary classification by taking the value of prediction scores into account, instead of simply discretize them into binary labels. The key idea is to evaluate the true positive rate and the false positive rate with varying classification thresholds to generate a receiver operating characteristic curve, or ROC curve, with the false-positive rate on the x-axis and true positive rate on the y-axis. The area under this curve is then used as the metric to evaluate the performance of the binary classification models.

![ROC curve example](Figure 2.2: An illustrative example for ROC curve. Figure from Wikipedia contributors (2022).)

An illustrative example is shown in Figure 2.2. The reader shall see that the AUC metric ranges from 0 to 1, with 0.5 being the score for a random classifier. The larger the AUC metric, the stronger the classifier.
CHAPTER 2. BACKGROUND

**F1-mult** Intuitively, we could simply use F1 as the evaluation metric when the prediction score can be discretized into binary labels. However, the task of quality estimation cares about correctly identifying the correctly and incorrectly predicted tokens equally, and since the datasets used for evaluation generally exhibit a label skew towards correctly predicted tokens, reporting a single F1 score could overestimate the model’s capability for identifying errors. Motivated by this problem, the WMT word-level quality estimation shared task 2013—2019 proposed to use the metric of **F1-mult**, which is simply multiplying the F1 score of OK and BAD labels.

**Matthews Correlation Coefficient (MCC)** Equipped with a similar capability in evaluating classification quality under a label skew, Matthews Correlation Coefficient (Matthews, 1975, MCC) has been used as the metric for the WMT word-level quality estimation shared task since 2020. This metric is formulated as follows:

\[
S = \frac{TP + FN}{N} \quad (2.27)
\]

\[
P = \frac{TP + FP}{N} \quad (2.28)
\]

\[
MCC = \frac{TP/N - S \times P}{\sqrt{PS(1 - S)(1 - P)}} \quad (2.29)
\]

where \(TP/FP\) stands for true/false positives and \(TN/FN\) stands for true/false negatives. \(N\) stands for the number of examples in the dataset.
CHAPTER 2. BACKGROUND

2.5 Summary

To summarize the methods and evaluations that we have covered in Section 2.3 and Section 2.4, we list some of the representative methods and evaluations in Table 2.1.
## Chapter 2. Background

### Table 2.1: Summary of the methods and evaluations we have covered in Section 2.3 and Section 2.4

<table>
<thead>
<tr>
<th>Methods</th>
<th>Explanation</th>
<th>Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gradient-based</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Saliency</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Simonyan et al., 2013)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• SmoothGrad</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Smilkov et al., 2017)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Integrated Gradients</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Sundararajan et al., 2017)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Pseudo Gradient-based</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• $\varepsilon$-LRP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Bach et al., 2015)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• DeepLIFT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Shrikumar et al., 2017)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Perturbation-based</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Occlusion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Zeiler and Fergus, 2014)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Shapley Value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Strumbelj and Kononenko, 2014)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Post-hoc Original</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• LIME</td>
<td></td>
<td>• Deep Ensemble</td>
</tr>
<tr>
<td>(Ribeiro et al., June 2016)</td>
<td></td>
<td>(Lakshminarayan et al., 2017)</td>
</tr>
<tr>
<td>• Word Aligners</td>
<td></td>
<td>• Meta-model</td>
</tr>
<tr>
<td>(Och and Ney, 2003)</td>
<td></td>
<td>(Chen et al., 2019)</td>
</tr>
<tr>
<td><strong>Post-hoc Dedicated</strong></td>
<td></td>
<td>• Rejection Option</td>
</tr>
<tr>
<td>• LIME</td>
<td></td>
<td>(Cortes et al., 2016)</td>
</tr>
<tr>
<td>(Ribeiro et al., June 2016)</td>
<td></td>
<td>• TrustScore</td>
</tr>
<tr>
<td>• Word Aligners</td>
<td></td>
<td>(Jiang et al., 2018)</td>
</tr>
<tr>
<td>(Och and Ney, 2003)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Intrinsic</strong></td>
<td></td>
<td>• Logistic Regression</td>
</tr>
<tr>
<td>• Logistic Regression</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• kNN</td>
<td></td>
<td>• ECE</td>
</tr>
<tr>
<td>• Attention</td>
<td></td>
<td>(Naeini et al., 2015)</td>
</tr>
<tr>
<td>(Bahdanau, Cho, and Bengio, 2014)</td>
<td></td>
<td>• Brier Score</td>
</tr>
<tr>
<td>• Attention</td>
<td></td>
<td>(Brier et al., 1950)</td>
</tr>
<tr>
<td>• Logistic Regression</td>
<td></td>
<td>• AUC</td>
</tr>
<tr>
<td>• kNN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Attention</td>
<td></td>
<td>• F1-mult</td>
</tr>
<tr>
<td>(Bahdanau, Cho, and Bengio, 2014)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Logistic Regression</td>
<td></td>
<td>• MCC</td>
</tr>
<tr>
<td>• kNN</td>
<td></td>
<td>(Matthews, 1975)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Evaluations</th>
<th><strong>Plausibility</strong></th>
<th><strong>Faithfulness</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>• Pointing Game</td>
<td>(Zhang et al., 2016)</td>
<td>(Alvarez-Melis and Jaakkola, 2018)</td>
</tr>
<tr>
<td>• Existing Human</td>
<td>annotations</td>
<td>• Human Evaluation</td>
</tr>
<tr>
<td>• Human Evaluation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Existing Human</td>
<td>annotations</td>
<td>• Human Evaluation</td>
</tr>
<tr>
<td>• Human Evaluation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Existing Human</td>
<td>annotations</td>
<td>• Human Evaluation</td>
</tr>
<tr>
<td>• Human Evaluation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Existing Human</td>
<td>annotations</td>
<td>• Human Evaluation</td>
</tr>
<tr>
<td>• Human Evaluation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Existing Human</td>
<td>annotations</td>
<td>• Human Evaluation</td>
</tr>
<tr>
<td>• Human Evaluation</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

63
Intended to be blank.
Part I

Explanation
Intended to be blank.
Chapter 3

A Framework for Evaluating

Explanations – Case Study on Neural Language Models

3.1 Introduction

In Chapter 2, we have defined the task of model explanation as assigning an importance score to each feature in the input feature set $F$, regarding a specific prediction $y$ made by a neural network model $M$. We have also reviewed many studies that uses explanation methods to uncover the neural network models’ internal decision mechanism. However, with
CHAPTER 3. A FRAMEWORK FOR EVALUATING EXPLANATIONS – CASE STUDY ON NEURAL LANGUAGE MODELS

Table 3.1: An example from our evaluation where different saliency methods assign different importance scores for the same model (Transformer language model) and the same next word prediction (*are*). G, SG and IG stands for gradient Saliency method, SmoothGrad and Integrated Gradients, respectively (see Section 2.3.2 for details). The tints of green and yellow mark the magnitude of positive and negative importance scores, respectively.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Importance Scores</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>U.S. companies wanting to expand in Europe</td>
<td></td>
</tr>
<tr>
<td>SG</td>
<td>U.S. companies wanting to expand in Europe</td>
<td></td>
</tr>
<tr>
<td>IG</td>
<td>U.S. companies wanting to expand in Europe</td>
<td></td>
</tr>
</tbody>
</table>

those insights uncovered, it should also be noted that different methods often give different explanations even when the internal decision mechanism remains the same (i.e., with $F$, $y$ and $M$ held constant), as exemplified in Table 3.1. Even so, most existing work that deploys these methods often makes an ungrounded assumption that a specific explanation method can reliably uncover the internal model decision mechanism. Some other studies rely merely on qualitative inspection to determine their applicability. Such practice has been pointed out in Adebayo et al. (2018), Lipton (2018), and Belinkov and Glass (2019) to be potentially problematic for model explanation studies – it can lead to misleading conclusions about the deep learning model’s reasoning process. On the other hand, in the context of NLP, the quantitative evaluation of model explanations largely remains an open problem (Belinkov and Glass, 2019).

In this chapter, we address this problem by building a comprehensive quantitative benchmark to evaluate model explanation methods. Our benchmark focuses on a fundamental category of NLP models: neural language models. Following the concepts proposed by
CHAPTER 3. A FRAMEWORK FOR EVALUATING EXPLANATIONS – CASE STUDY ON NEURAL LANGUAGE MODELS

Jacovi and Goldberg (2020), our benchmark evaluates the credibility of model explanations from two aspects: plausibility and faithfulness. In short, plausibility measures how much these explanations align with basic human intuitions about the model decision mechanism, while faithfulness measures how consistent the explanations are regarding perturbations that are supposed to preserve the same model decision mechanism on either the input feature $F$ or the model $M$.

With these concepts in mind, our main contribution is materializing these tests’ procedure in the context of neural language modeling and building four test datasets from existing linguistic annotations to conduct these tests. On those datasets, we evaluated some of the most commonly used post-hoc model explanation methods, based on state-of-the-art language models on three different network architectures. Our evaluation reveals that these model explanation methods’ applicability depends heavily on specific choices of explanation methods, model architectures, and model configurations. We suggest that future work deploying these methods to NLP models should carefully validate their explanations before drawing conclusions.

This chapter is organized as follows: Section 3.2 describes the plausibility and faithfulness tests in our evaluation; Section 3.3 presents the datasets we built for the evaluation; Section 3.5 presents our experiment setup and results; Section 3.6 discusses some limitations and implications of the evaluation; Section 3.7 concludes the chapter.
3.2 Evaluation Paradigm

In this section, we first introduce the notion of plausibility and faithfulness in the context of neural network explanations (following Jacovi and Goldberg (2020)), and then introduce the test we adopt to evaluate them respectively.

3.2.1 Plausibility

Concept  An explanation is plausible if it aligns with human intuitions about how a specific neural model makes decisions. For example, intuitively, an image classifier can identify the object in the image because it can capture some features of the main object in the image. Hence, a plausible explanation would assign high importance to the area occupied by the main object. This idea of comparison with human-annotated ground-truth (often as “bounding-boxes” signaling the main object’s area) is used by various early studies in computer vision to evaluate model explanation methods’ reliability (Jiang et al., 2013, *inter alia*). However, the critical challenge of such evaluations for neural language models is the lack of such ground-truth annotations.

Test  To overcome this challenge, we follow Poerner, Schütze, and Roth (2018) to construct ground-truth annotations from existing lexical agreement annotations. Consider, for example, the case of morphological number agreement. Intuitively, when the language model predicts
a verb with a singular morphological number, the singular nouns in the prefix should be considered important features, and vice versa. Based on this intuition, we divide the nouns in the prefix into two different sets: the cue set $C$, which shares the same morphological number as the verb in the sentence; and the attractor set $A$, which has a different morphological number than the verb in the sentence.

Then, according to the prediction $y$ made by the model $M$, the test will be conducted under one of the two following scenarios:

- **Expected**: when $y$ is the verb with the correct number, the explanation passes the test if $\max_{w \in C} \psi(w) > \max_{w \in A} \psi(w)$

- **Alternative**: when $y$ is the verb with the incorrect number, the explanation passes the test if $\max_{w \in C} \psi(w) < \max_{w \in A} \psi(w)$

**Interpreted Model** However, for some post-hoc explanation methods, the test described above has a flaw: while the evaluation criteria focus on a specific category of lexical agreement, the prediction of a word could depend on multiple lexical agreements simultaneously. To illustrate this point, consider the verb prediction following the prefix “At the polling station people ...”. Suppose the model $M$ predicts the verb vote. One could argue that people is more important than polling station because it needs the subject to determine the morphological number of the verb. However, the semantic relation between vote and polling station is also important because that is what makes vote more likely than other random
verbs, e.g. *sing*.

To minimize such discrepancy and constrain the scope of agreements used to make predictions, we draw inspiration from the previous work on representation probing and make adjustment to the model \( M \) we are evaluating on (Tenney, Das, and Pavlick, 2019; Tenney et al., 2019; Kim et al., 2019; Conneau et al., 2018; Adi et al., 2017; Shi, Padhi, and Knight, 2016). The idea (illustrated in Figure 3.1) is to take a language model that is trained to predict words (e.g., *vote* in the example above) and substitute the original final linear layer with a new linear layer (which we refer to as a *probe*) finetuned to predict a binary lexical agreement tag (e.g., *PLURAL*) corresponding to the word choice. By making this adjustment, the final layer extracts a subspace in the representation that is relevant to the prediction of particular lexical agreement during the forward computation, and reversely.

Figure 3.1: Illustration of language model prediction head replacement for model explanation evaluation. All language models are first trained with the word prediction head following the standard language model training recipe, then a probe prediction head is finetuned on synthetic finetuning data and used for evaluations of model explanation methods.
CHAPTER 3. A FRAMEWORK FOR EVALUATING EXPLANATIONS – CASE STUDY ON NEURAL LANGUAGE MODELS

filters out gradients that are irrelevant to the agreement prediction in the backward pass, creating an explanation that is only subject to the same agreement constraints as to when the annotation for the test set is done.

Apart from the adjustment made on the model \( M \) above, we also extend Poerner, Schütze, and Roth (2018) in the other two aspects: (1) we evaluate on one more lexical agreement: gender agreements between pronouns and referenced entities, and on both natural and synthetic datasets; (2) instead of evaluating on small models, we evaluate on large state-of-the-art models for each architecture. We also show that evaluation results obtained on smaller models cannot be trivially extended to larger models.

3.2.2 Faithfulness

Concept An explanation is faithful if the feature importance it assigns is consistent with the internal decision mechanism of a model. However, as Jacovi and Goldberg (2020) pointed out, the notion of “decision mechanism” lacks a standard definition and a practical way to make a comparison. Hence, as a proxy, we follow the working definition of faithfulness as proposed in their work, which states that an explanation is faithful if the feature importance it assigns remains consistent with changes that should not change the internal model decision mechanism. Among the three relevant factors for model explanation methods (prediction \( y \), model \( M \), and input feature set \( F \)), we focus on consistency upon changes in model
CHAPTER 3. A FRAMEWORK FOR EVALUATING EXPLANATIONS – CASE STUDY ON NEURAL LANGUAGE MODELS

$M$ (model consistency) and input feature set $F$ (input consistency).\(^1\) Note that these two consistencies respectively correspond to assumptions 1 and 2 in the discussion of faithfulness evaluation in Jacovi and Goldberg (2020).

**Model Consistency Test**  To measure model consistency, we propose to measure the consistency between feature importance $\Psi_M(F)$ and $\Psi_M'(F)$, which are generated by two different models $M$ and $M'$ that has a very similar internal decision mechanism. Thus, if the model explanation method is being faithful to the internal decision mechanism, $\Psi_M(F)$ and $\Psi_M'(F)$ should have high consistency.

The question is how to build different $M$ and $M'$ and yet maintain similar internal decision mechanism. In our case study, because we are also interested in analyzing the interaction between model size and explanation quality, we choose to build a smaller $M'$ from the larger model $M$ by performing knowledge distillation (Hinton, Vinyals, and Dean, 2015; Sanh et al., 2019) from $M$, which trains $M'$ by forcing its output distribution to be similar to that of $M$. While there is no formal guarantee that $M$ and $M'$ will have the entirely same internal decision mechanism (not even $M$ and $M'$ have the same model size), our hope is that such setup will provide a “best-effort” proxy to make comparisons on faithfulness without being constrained to the input words in the cue/attractor set.

\(^1\) Although evaluating explanation consistency over similar predictions $y$ is also possible, it is not of interest as most applications expect different explanations for different predictions.

74
CHAPTER 3. A FRAMEWORK FOR EVALUATING EXPLANATIONS – CASE
STUDY ON NEURAL LANGUAGE MODELS

**Input Consistency Test** To measure input consistency, we perform substitutions in the
input and measure the consistency between feature importance $\Psi(F)$ and $\Psi(F')$, where
$F$ and $F'$ are input features sets before/after the substitution. For example, the following
prefix-prediction pairs should have the same feature importance distribution:

- *The nun bought the son a gift because (she...)*
- *The woman bought the boy a gift because (she...)*

For all faithfulness evaluations, we measure consistency by measuring Pearson correla-
tion between pairs of importance distributions, which means $\Psi_M(F)$ and $\Psi_{M'}(F')$ for model
consistency test and $\Psi(F)$ and $\Psi(F')$ for input consistency test. We collect instance-level
statistics and aggregate them over the test set to compute system-level Pearson correlation,
which serves as our evaluation metric for faithfulness evaluation. Also, note that although
we can theoretically conduct faithfulness tests with any model $M$ and any dataset with a
feasible substitution scheme, for the simplicity of analysis and data creation, we will use the
same model $M$ (with lexical agreement probes) and the same dataset as plausibility tests.

### 3.3 Data

Following the formulation in Section 3.2, we constructed four datasets for our benchmark,
as exemplified in Table 3.2. Two of the datasets are concerned with *number agreement* of a
Table 3.2: Examples prefixes from the four evaluation datasets, followed by the probing tag prediction under the expected scenario. The cue and attractor sets are marked with solid Green and yellow, respectively.

verb with its subject. The other two are concerned with gender agreement of a pronoun with its antecedent entity mentions. For each lexical agreement type, we have one synthetic dataset and one natural dataset. Both synthetic datasets ensure there is only one cue and one attractor for each test instance, while for natural datasets, there are often more than one.

For number agreement, our synthetic dataset is constructed from selected sections of Syneval, a targeted language model evaluation dataset from Marvin and Linzen (2018), where the verbs and the subjects could be easily induced with heuristics. We only use the most challenging sections where strongly interceding attractors are involved. Our natural dataset for this task is filtered from Penn Treebank (Marcus, Santorini, and Marcinkiewicz, 1993, PTB), including training, development, and test. We choose PTB because it offers not only human-annotated POS-tags necessary for benchmark construction but also dependent subjects of verbs for further analysis.
For gender agreement, our synthetic dataset comes from the unambiguous Winobias coreference resolution dataset used in Jumelet, Zuidema, and Hupkes (2019), and we only use the 1000-example subset where there is respectively one male and one female antecedent. Because this dataset is intentionally designed such that most humans will find pronouns of either gender equally likely to follow the prefix, no such pronoun gender is considered to be “correct”. Hence, without loss of generality, we assign the female pronoun to be the expected case.2 Our natural dataset for this task is filtered from CoNLL-2012 shared task dataset for coreference resolution (Pradhan et al., 2012, also including training, development, and test). The prefix of each test example covers a document-level context, which usually spans several hundred words.

Plausibility Test For number agreement, the cue set $C$ is the set of all nouns that have the same morphological number as the verb. In contrast, the attractor set $A$ is the set of all nouns with a different morphological number. For gender agreement, the cue set $C$ is the set of all nouns with the same gender as the pronoun, while the attractor set $A$ is the set of all nouns with a different gender.

Model Consistency Test No special treatment to data is needed for this test. We conduct model consistency tests on all datasets we built.

---

2Note that this assumption will not change the explanations we generate or the benchmark test conducted for explanations, as we always explain the argmax decision of the model, which is not affected by this assumption. It will only affect the breakdown of the result we report.
CHAPTER 3. A FRAMEWORK FOR EVALUATING EXPLANATIONS – CASE STUDY ON NEURAL LANGUAGE MODELS

**Input Consistency Test** We recognize that generating explanation-preserving input perturbations for natural datasets is quite tricky. Hence, unlike the model consistency test, we focus on the two synthetic datasets for faithfulness tests because they are generated from templates. As can be seen from the examples, when the nouns in the cue/attractor set are substituted while maintaining the lexical agreement, the underlying model decision mechanism should be left unchanged; hence they can be viewed as explanation-preserving perturbations. We identified 24 and 254 such explanation-preserving templates from our Syneval and Winobias dataset and generated perturbations pairs by combining the first example of each template with other examples generated from the same template.

### 3.4 Composition Scheme

So far, our discussion has been focusing on a general evaluation framework for explanation methods without making assumptions for a specific method. Before discussing our case study on some post-hoc model explanation methods, we would like to discuss an issue that specifically arises when applying post-hoc model explanation methods to NLP models: the composition scheme. Recall that in Section 2.3.2, we mentioned that the necessity of composition scheme is driven by the need to compose a vector of importance scores $\tilde{\psi}$ over dimensions of the word embedding into a scalar token-level importance $\psi$. An existing proposal from Li et al. (2016) composes the scores by calculating 1-norm of the
CHAPTER 3. A FRAMEWORK FOR EVALUATING EXPLANATIONS – CASE STUDY ON NEURAL LANGUAGE MODELS

vector (Equation 2.16). In the next chapters, we will refer to this composition scheme as Vector Norm (VN) scheme. We argue that there is a more mathematically principled way to approach this composition operation.

To start, we treat the word embedding look-up operation as a dot product between the embedding weight matrix $E$ and an one-hot vector $z \in \{0, 1\}^{|V|}$. When mapping a word $y_j$ into its embedding vector $e_j$, we can construct the corresponding one-hot vector $z_j$ by only setting the entry corresponding to $y_j$ as 1 and leaving the rest as 0. In this way, the mapping operation could be transformed into a matrix-vector multiplication:

$$e_j = E \cdot z_j \quad (3.1)$$

Similarly, for each token in the sentence prefix input $s_j = y_0y_1 \ldots y_{j-1}$ to the language model, we could construct a one-hot vector $z_0, z_1, \ldots, z_{j-1}$.

Suppose we relax $z$ from a discrete vector $\{0, 1\}^{|V|}$ into a continuous vector $\mathbb{R}^{|V|}$. We can similarly define the importance of each element for some $z_k$ with regard to the prediction $y_j$ as$^3$:

$$\tilde{\psi}(z_k) = \left[ \frac{\partial p(y_j | s_j)}{\partial z_k} \right]^T \quad (3.2)$$

$^3$The transpose is added to ensure $\tilde{\psi}(z_k)$ is a column vector, which is useful for the subsequent derivations.
with $0 \leq k < j$. Note that in the computation graph, the language model probability $p(y_j \mid s_j)$ is dependent on the word embeddings of the words in its prefix $e_0, e_1, \ldots, e_{j-1}$, which in turn are dependent on $z_0, z_1, \ldots, z_{j-1}$ Hence $\tilde{\psi}(z_k)$ is not trivially 0.

Now, we consider what this new element-wise importance score $\tilde{\psi}$ stands for. It appears that we can treat the 1 in the one-hot vector $z_k$ as the identity of the word $y_k$. Hence, we propose to use the element-wise importance score of the only 1-entry in $z_k$ as the importance score of the word $y_k$ and simply discard the scores of the 0-entries. This is equivalent to performing a dot product with $z_k$:

$$\psi(e_k) = \frac{\tilde{\psi}(z_k)^T \cdot z_k}{\sum_{k=0}^{j-1} \tilde{\psi}(z_k)^T \cdot z_k} \tag{3.3}$$

We now look at how $\psi(e_k)$ relates to the gradients of the word embeddings $\frac{\partial p(y_j \mid s_j)}{\partial e_k}$. For simplicity, let’s skip the normalization term in the denominator and focus on the nominator:

$$\tilde{\psi}(z_k)^T \cdot z_k = \frac{\partial p(y_j \mid s_j)}{\partial z_k} \cdot z_k$$
$$= \frac{\partial p(y_j \mid s_j)}{\partial e_k} \cdot \frac{\partial e_k}{\partial z_k} \cdot z_k$$
$$= \frac{\partial p(y_j \mid s_j)}{\partial e_k} \cdot E \cdot z_k$$
$$= \frac{\partial p(y_j \mid s_j)}{\partial e_k} \cdot e_k \tag{3.4}$$
CHAPTER 3. A FRAMEWORK FOR EVALUATING EXPLANATIONS – CASE STUDY ON NEURAL LANGUAGE MODELS

In other words, \( \tilde{\psi}(z_k)^T \cdot z_k \) is simply the directional derivative of the gradient \( \frac{\partial p(y_j|s_j)}{\partial e_k} \) along the direction of the word embedding vector \( e_k \). This is convenient, because word embedding look-up is never implemented as a matrix multiplication, but rather as a table look-up. This means that for each input word \( y_k \), there is no \( z_k \) vector we can derive gradient for. Equation 3.4 points out that the quantity defined in Equation 3.3 can actually be computed without creating \( z_k \) or involving the whole embedding matrix \( E \).

In the existing literature, a similar derivation was done in Denil, Demiraj, and Freitas (2014), but for retrieving sentence embeddings from a document. Note that this is different from the “Input \( \times \) Gradient” method proposed in Shrikumar et al. (2016) or “Gradient \( \odot \) Input” in Kindermans et al. (2019). Shrikumar et al. (2016) defines an alternative back-propagation rule the happens at every level of the network, while our Gradient \( \cdot \) Input operation happens after the gradient is computed. Kindermans et al. (2019) performs an element-wise product between two matrices, while we perform a dot product between two vectors.

We conducted preliminary experiments for both the Vector Norm composition scheme (Li et al., 2016) and the Gradient \( \cdot \) Input composition scheme. Results from these experiments showed that the Gradient \( \cdot \) Input composition scheme significantly outperform the Vector Norm composition scheme. Hence, apart from discussion of those preliminary results in Section 3.5.3, all the results we report in this chapter are using the Gradient \( \cdot \) Input
CHAPTER 3. A FRAMEWORK FOR EVALUATING EXPLANATIONS – CASE STUDY ON NEURAL LANGUAGE MODELS

composition scheme.

3.5 Experiments

3.5.1 Setup

Interpretation Methods Our case study covers three gradient-based post-hoc model explanation methods we covered in Chapter 2: the gradient saliency method (Simonyan, Vedaldi, and Zisserman, 2013, G), SmoothGrad (Smilkov et al., 2017, SG), and Integrated Gradients (Sundararajan, Taly, and Yan, 2017, IG). When implementing SmoothGrad for Gradient · Input composition scheme, we did not fully follow the task definition and perturb the 1-entry in \( z_k \), i.e., multiply a scalar perturbation factor on the word embedding \( e_k \) corresponding to a token \( x_k \). Rather, we directly sample a random noise vector \( \sigma_k \) from normal distribution that has the same dimensions as \( e_k \) and add it to \( e_k \). This modification allows us to introduce more degrees of freedom for perturbations of \( e_k \).

Another minor complication is that our input tokens are segmented by subword segmentation methods like Byte-Pair Encoding or sentencepiece, depending on the preprocessing used by individual models. However, the human annotations on our evaluation dataset were constructed on word-level, and the evaluation results cannot be easily compared across different subword segmentation schemes. Hence, our evaluation are always performed on
word-level feature importance distributions, and subword-level distributions are converted to word-level by simply adding up the importance score of each subword corresponding to a word.

As for hyperparameters, for SG, we set sample size $N = 30$ and sample standard error $\sigma$ to be $0.15 \cdot (\max(E_s) - \min(E_s))^4$; for IG, we use step size $N = 100$. These choices are made by referring to the original papers (Smilkov et al., 2017; Sundararajan, Taly, and Yan, 2017) and verified on a small held-out development set.

**Interpreted Model** Our case study covers three different neural language model architectures, namely LSTM (Hochreiter and Schmidhuber, 1997), QRNN (Bradbury et al., 2017) and Transformer (Vaswani et al., 2017; Baevski and Auli, 2019; Dai et al., 2019). All language models are trained on WikiText-103 dataset (Merity et al., 2017). For the first two architectures, we use the implementation as in awd-lstm-lm toolkit (Merity, Keskar, and Socher, 2018). For Transformer, we use the implementation in fairseq toolkit (Ott et al., 2019).

For all the task-specific “probes”, the finetuning is performed on examples extracted from WikiText-2 training data. A tuning example consists of an input prefix and a gold tag for the lexical agreement in both cases. For number agreement, we first run Stanford POS Tagger (Toutanova et al., 2003) on the data, and an example is extracted for each

$E_s = [e_0, e_1, \ldots, e_j]$. In other words, $E_s$ is the queried embedding matrix as the input to the network at time step $j$. $\max$ and $\min$ stand for the maximum and the minimum element respectively.
present tense verb and each instance of was or were. For gender agreement, an example is extracted for each gendered pronoun. During finetuning, we fix all the other parameters except the final linear layer. The final layer is tuned to minimize cross-entropy, with Adam optimizer (Kingma and Ba, 2015) and initial learning rate of $1e^{-3}$ with ReduceLROnPlateau scheduler.

We follow the setup for DistilBERT (Sanh et al., 2019) for the distillation process involved during the model consistency test, which reduces the depth of models but not the width. The training objective is in three folds:

- $L_{ce}$: the cross entropy between the output distribution of the parent model $M$ and the child model $M'$

- $L_{lm}$: the normal loss function used to train the language models

- $L_{cos}$: the cosine similarity between the final-layer hidden states of the parent model $M$ and the child model $M'$

The final loss function is simply an addition of the individual components above.

For our LSTM (3 layers) and QRNN model (4 layers), the $M'$ we distill is one layer shallower than the original model $M$. For our transformer model (16 layers), we distill a 4-layer $M'$ largely due to memory constraints.
CHAPTER 3. A FRAMEWORK FOR EVALUATING EXPLANATIONS – CASE
STUDY ON NEURAL LANGUAGE MODELS

<table>
<thead>
<tr>
<th></th>
<th>Number Agreement</th>
<th>Gender Agreement</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>all</td>
<td>PTB</td>
</tr>
<tr>
<td>Random</td>
<td>0.546</td>
<td>0.454</td>
</tr>
<tr>
<td>Nearest</td>
<td>0.502</td>
<td>0.498</td>
</tr>
<tr>
<td>LSTM G</td>
<td>0.452</td>
<td>0.484</td>
</tr>
<tr>
<td>LSTM SG</td>
<td>0.780</td>
<td>0.805</td>
</tr>
<tr>
<td>LSTM IG</td>
<td>0.816</td>
<td>0.856</td>
</tr>
<tr>
<td>QRNN G</td>
<td>0.463</td>
<td>0.501</td>
</tr>
<tr>
<td>QRNN SG</td>
<td>0.575</td>
<td>0.599</td>
</tr>
<tr>
<td>QRNN IG</td>
<td>0.697</td>
<td>0.728</td>
</tr>
<tr>
<td>Transformer G</td>
<td>0.551</td>
<td>0.551</td>
</tr>
<tr>
<td>Transformer SG</td>
<td>0.842</td>
<td>0.851</td>
</tr>
<tr>
<td>Transformer IG</td>
<td>0.734</td>
<td>0.741</td>
</tr>
</tbody>
</table>

Table 3.3: Plausibility case study result. Each number is the fraction of cases the explanation passes the benchmark test, while the numbers in brackets for each architecture are the fraction of times these scenarios occur for predictions generated by the corresponding model. Results from the best explanation method for each architecture are boldfaced. The exp. and alt. columns are breakdown of evaluation results into expected scenarios and alternative scenarios as defined in Section 3.2. G, SG, IG stands for the gradient saliency, SmoothGrad, and Integrated Gradients, respectively.

### 3.5.2 Main Results

**Plausibility** According to our plausibility evaluation result, summarized in Table 3.3, both SG and IG consistently perform better than the gradient saliency method regardless of different benchmark datasets and explained models. However, the comparison between SG and IG explanations varies depending on the model architecture and test sets.

Across different architectures, the methods achieve the best plausibility for the Transformer language model except on the Syneval dataset. LSTM closely follows Transformer
## Table 3.4: Faithfulness Benchmark Result

Each number is the average Pearson correlation computed on the corresponding dataset. Results from the best explanation method for each architecture are boldfaced. Refer to the caption of Table 3.3 for other notations.
for most benchmarks, while the plausibility of the explanation from QRNN is much worse. Another trend worth noting is that the gap between Transformer and the other two architectures is much larger on the CoNLL benchmark, which is the only test that involves explaining document-level contexts. However, these architectures' prediction accuracy is similar, meaning that there is no significant modeling power difference for gender agreements in this dataset. We hence conjecture that the recurrent structure of LSTM and QRNN might diminish gradient signals with increasing time steps, which causes the deterioration of explanation quality for long-distance agreements – a problem that Transformer is exempt from, thanks to the self-attention structure.

**Faithfulness**  Table 3.4a shows the input consistency benchmark result. Firstly, it can be seen that the explanations of LSTM and Transformer are more resilient to input perturbations than that of QRNN. This is the same trend as we observed for plausibility benchmark on these datasets. When comparing different model explanation methods, we see that SG consistently outperforms for Transformer, but fails for the other two architectures, especially for QRNN. Also, note that achieving higher plausibility does not necessarily imply higher faithfulness. For example, compared to the gradient saliency method, SG and IG almost always significantly improve plausibility but do not always improve faithfulness. This lack of improvement is different from the findings in computer vision (Yeh et al., 2019a), where they show both SG and IG improve input consistency. Also, for LSTM, although SG
works slightly better than IG in terms of plausibility, IG outperforms SG in terms of input consistency by a large margin.

Table 3.4b shows the model consistency benchmark result. One should first notice that model consistency numbers are lower than input consistency across the board, and the drop is more significant for LSTM and QRNN even though their student model is not as different as the Transformer model (<20% parameter reduction vs. 61%). As a result, there is a significant performance gap in terms of best model consistency results between LSTM/QRNN and Transformer. Note that, like in plausibility results, such gap is most notable on the CoNLL dataset. On the other hand, when comparing between model explanation methods, we again see that SG outperforms for Transformer while failing most of the times for QRNN and LSTM.

3.5.3 Analysis

Vector Norm Composition Scheme

In this section, we report the preliminary results we obtained with VN composition scheme. We would like to argue first that even mathematically, VN is not a good fit for our evaluation paradigm. Vector norm composition scheme will only indicate the importance of a feature, but will not indicate the polarity of the importance because it cannot generate a negative word importance score. The notion of polarity is important
because our plausibility evaluation does distinguish between input words that should have positive/negative importance scores by placing them in cue and attractor sets, respectively.

For example, in Table 3.1, the singular proper noun U.S. and Europe are important input words because they could potentially lead the model to make the alternative prediction is instead of the expected prediction are. Hence, they are placed into the attractor set, and when explaining the next word prediction are, our plausibility test expects that they should have large negative importance scores.

Our preliminary results for VN composition scheme are shown in Table 3.5. For the gradient saliency method, the VN composition scheme performs on-par with the Gradient ∙ Input scheme (see Table 3.3). However, with SmoothGrad, the plausibility result does not significantly improve like the case with the Gradient ∙ Input scheme, which makes its performance trail behind by a large margin.
CHAPTER 3. A FRAMEWORK FOR EVALUATING EXPLANATIONS – CASE STUDY ON NEURAL LANGUAGE MODELS

With these theoretical and empirical evidence, we decided to drop vector norm composition scheme for the rest of our analysis.

<table>
<thead>
<tr>
<th></th>
<th>Model Type</th>
<th>Example 1</th>
<th>Example 2</th>
<th>Example 3</th>
<th>Example 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a</td>
<td>QRNN+SG</td>
<td>The [grandmother] examined the (grandson) for injuries because</td>
<td>The [sister] examined the (groom) for injuries because</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1b</td>
<td>QRNN+SG</td>
<td>The [grandmother] examined the (grandson) for injuries because</td>
<td>The [sister] examined the (groom) for injuries because</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2a</td>
<td>QRNN+G</td>
<td>The [grandmother] examined the (grandson) for injuries because</td>
<td>The [aunt] examined the (groom) for injuries because</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2b</td>
<td>QRNN+G</td>
<td>The [grandmother] examined the (grandson) for injuries because</td>
<td>The [aunt] examined the (groom) for injuries because</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3a</td>
<td>QRNN+SG</td>
<td>The [grandmother] examined the (grandson) for injuries because</td>
<td>The [aunt] examined the (groom) for injuries because</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3b</td>
<td>QRNN_distilled+SG</td>
<td>The [grandmother] examined the (grandson) for injuries because</td>
<td>The [aunt] examined the (groom) for injuries because</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4a</td>
<td>Transformer+SG</td>
<td>The [grandmother] examined the (grandson) for injuries because</td>
<td>The [aunt] examined the (groom) for injuries because</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4b</td>
<td>Transformer+SG</td>
<td>The [grandmother] examined the (grandson) for injuries because</td>
<td>The [aunt] examined the (groom) for injuries because</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4c</td>
<td>Transformer_distilled+SG</td>
<td>The [grandmother] examined the (grandson) for injuries because</td>
<td>The [aunt] examined the (groom) for injuries because</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3.6: Examples from Winobias dataset for qualitative analysis. Cue words are marked with [] while attractor words are marked with (). The tints of green and yellow mark the magnitude of positive and negative importance scores, respectively. For all examples, the prediction interpreted is the FEMININE tag. 1 is a case with high plausibility and low input faithfulness; 2 is a case with low plausibility and high input faithfulness; 3 is a case with low model faithfulness; 4 is a case with high plausibility and high input/model faithfulness.

**Plausibility vs. Faithfulness** A natural question for our evaluation is how the property of plausibility and faithfulness interact with each other. Table 3.6 illustrates such interaction with examples. Among them, 1 and 2 are two cases where the plausibility and input faithfulness evaluation results do not correlate. In general, the explanations in both cases are of low quality, but they also fail in different ways. In case 1, the explanation assigns the correct relative ranking for the cue words and attractor words, but the importance of the words outside the cue/attractor set varies upon perturbation. On the other hand, in case 2, the importance ranking among features is roughly maintained upon perturbation, but the importance score assigned for both examples do not agree with the prediction explained...
(FEMININE tag) and thus can hardly be understood by humans. It should be noted that these defects can only be revealed when both plausibility and faithfulness tests for explanations are deployed.

Case 3 shows a scenario where the model explanation yields very different explanations for the same input/prediction pair, indicating that explanations from this architecture/explanation method combination are subject to changes upon changes in the architecture configurations. Finally, in case 4, we see that an architecture/explanation method combination performing well in all tests yields stable explanations that humans can easily understand.

<table>
<thead>
<tr>
<th></th>
<th>Syneval</th>
<th>Winobias</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>all</td>
<td>exp. alt.</td>
</tr>
<tr>
<td>best plausibility</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LSTM (SG)</td>
<td>0.945</td>
<td>0.922</td>
</tr>
<tr>
<td>QRNN (IG)</td>
<td>0.981</td>
<td>0.964</td>
</tr>
<tr>
<td>Transformer (SG)</td>
<td>0.917</td>
<td>0.908</td>
</tr>
<tr>
<td>best (input)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>faithfulness</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LSTM (IG)</td>
<td>–</td>
<td>0.628</td>
</tr>
<tr>
<td>QRNN (IG)</td>
<td>–</td>
<td>0.733</td>
</tr>
<tr>
<td>Transformer (SG)</td>
<td>–</td>
<td>0.569</td>
</tr>
</tbody>
</table>

Table 3.7: Plausibility & input faithfulness on synthetic datasets with distilled models. Only results for the interpretation method with best performance are shown. Refer to the caption of Table 3.3 for other notations.

**Sensitivity to Model Configurations**  Our model faithfulness evaluation shows that variations in the model configurations (number of layers) could drastically change the model
CHAPTER 3. A FRAMEWORK FOR EVALUATING EXPLANATIONS – CASE STUDY ON NEURAL LANGUAGE MODELS

explanation in many cases. Hence, we want to answer two analysis questions: (1) are these explanations changing for the better or worse quality-wise with the distilled smaller models? (2) are there any patterns for such changes? For question (1), our comparison of the evaluation results from the parent and the child model is shown in Table 3.7. Overall, compared to the corresponding results in Table 3.3 (plausibility) and Table 3.4a (input faithfulness), the model explanation methods we evaluated perform better with the smaller distilled models. Most remarkably, we see a drastic performance improvement for QRNN, both in plausibility and faithfulness. For LSTM and Transformer, we observe an improvement for input faithfulness on Winobias and roughly the same performance for other tests.

Figure 3.2: Analysis of model configuration vs. plausibility on PTB and CoNLL benchmark. Each model configuration is color-coded, while the parameter size (in millions) is shown with circle size. \( l, w, e, h \) stands for model depth, width of feed-forward layers after self-attention, embedding size, and the number of heads.

As for the second question, we build smaller Transformer language models with various
CHAPTER 3. A FRAMEWORK FOR EVALUATING EXPLANATIONS – CASE STUDY ON NEURAL LANGUAGE MODELS

Figure 3.3: Analysis of model configuration vs. plausibility and faithfulness on Syneval benchmark. Each model configuration is color-coded, while the parameter size (in millions) is shown with circle size. $l$, $w$, $e$, $h$ stands for model depth, width of feed-forward layers after self-attention, embedding size, and the number of heads. Note that the faithfulness numbers plotted here are the ones explained with expected scenario predictions.

Depth, number of heads, embedding size, and feed-forward layer width settings, while keeping other hyperparameters unchanged. We show two different groups of comparison here. Figure 3.2 shows our investigation on the interaction of explanation plausibility on PTB and CoNLL test sets. In general, IG works better for deeper models, while SG works better for shallower models on the PTB test set, but remains roughly the same performance for all architectures on the CoNLL test set. This indicates the noisiness of the trend we are investigating, as both explanation methods and evaluation dataset choice can influence...
the trend. As for the other factors of the model configurations, the trend is even noisier (note how much rankings of different configurations change moving from shallow to deep models) and do not show any clear patterns.

Figure 3.3, on the other hand, focuses on one specific dataset and investigates the trend on both the plausibility and input faithfulness with varying model configurations. For plausibility results, we largely see the same trend as on PTB dataset. For faithfulness results, the trend for SG is largely the same as plausibility. For IG, the variance across other factors of configurations tends to be different on shallower models vs. deeper models, but overall still shows higher numbers for deeper models similar to the trend for plausibility.

Overall, these analyses show that the explanation qualities are sensitive to model configuration changes, and the trends of explanation quality on a specific model configuration cannot be easily generalized to other configurations.

Table 3.8: A number agreement test case where the distilled Transformer model makes the correct prediction (singular) but all explanation methods unanimously point to a singular noun that is not grammatical subject as the most salient cue for this prediction.

<table>
<thead>
<tr>
<th></th>
<th>The [fact] that this happened two (years) ago and</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>there was a [recovery]</td>
<td>SG</td>
<td>there was a [recovery]</td>
</tr>
<tr>
<td>IG</td>
<td>there was a [recovery]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Saliency vs. Probing** Our evaluation incorporates probing to focus only on specific lexical agreements of interest. It should be pointed out that in the literature of representation
probing, the method has always been working under the following assumption: when the model makes an expected-scenario ("correct") prediction, it has acquired the adequate linguistic knowledge to refer to a grammatical cue, for example, the subject of the verb in the number agreement case. However, in our evaluation, we also observe some interesting phenomena in the model explanations that breaks the assumption, which is exemplified in Table 3.8. This calls for future work that aims to better understand language model behaviors by examining other possible cues used for predictions made in representation probing under the validated cases where model explanation methods could be reliably applied.

3.6 Discussion

Most existing work on evaluating model explanation methods focuses only on computer vision models (Adebayo et al., 2020; Hooker et al., 2019; Adebayo et al., 2018; Heo, Joo, and Moon, 2019; Ghorbani, Abid, and Zou, 2019, inter alia). In the context of NLP, Poerner, Schütze, and Roth (2018) is the first work to conduct such evaluations for NLP and the only prior work that conducts such evaluations for neural language models but has several limitations as we have already pointed out in Section 3.2. Arras et al. (2019), Atanasova et al. (2020), and Hao (2020) conducted similar evaluations based on specifically designed diagnostic toy tasks and/or text classification, while Bastings and Filippova (2020) casted doubt on whether these conclusions could be generalized to sequence generation
CHAPTER 3. A FRAMEWORK FOR EVALUATING EXPLANATIONS – CASE STUDY ON NEURAL LANGUAGE MODELS

tasks. Li et al. (2020) evaluated various explanation methods for neural machine translation models by building proxy models on only the top-\(k\) important input words as determined by the explanation methods, but such evaluation requires generating explanations for a large training set and hence is intractable for even mildly computationally-expensive methods such as SmoothGrad and Integrated Gradients. On a slightly different line, DeYoung et al. (2020) built a benchmark to evaluate a specific category of NLP models that generate rationales during predictions, which is a different path towards building explainable NLP models.

Our evaluation is not without its limitations. The first limitation, inherited from earlier work by Poerner, Schütze, and Roth (2018), is that our plausibility test only concerns the words in cue/attractor sets rather than other words in the input prefix. Such limitation is inevitable because the annotations from which we build our ground-truth explanations are only concerned with a specific lexical agreement. In the next chapter, we will see that for some other applications, it is possible to perform plausibility evaluations that covers explanations for all predictions.

The second limitation is that the model consistency test is based on the assumption that the teacher model \(M\) and the distilled model \(M'\) will have very similar internal decision mechanism, which could be tenuous depending on the model distillation setup. Some more recent studies proposed alternative approaches such as erasure (Madsen et al., 2021) and counterfactual reasoning (Ge et al., 2021), which do not have dependency to our distillation-
related assumption above. To this day, the evaluation of explanation faithfulness is still an actively-researched open problem (Ge et al., 2021; Chan et al., 2022).

The third limitation is that the test sets used in these benchmarks need to be constructed in a case-to-case manner, according to the chosen lexical agreements and the input perturbations. While it is hard to create plausibility test sets without human interference, future work could explore automatic input consistency tests by utilizing adversarial input generation techniques in NLP (Alzantot et al., 2018; Cheng, Jiang, and Macherey, 2019; Cheng et al., 2020b).

It should also be noted that while our work focuses on evaluating a specific category of explanation methods for neural language models, our evaluation paradigm can be easily extended to evaluating other explanation methods such as attention mechanism, and with other sequence models such as masked language models (e.g., BERT). We would also like to extend these evaluations beyond English datasets, especially to languages with richer morphological inflections.

3.7 Conclusion

We build upon the properties of plausibility and faithfulness of a model explanation method to design an evaluation framework for model explanations on neural language models. Our evaluation dataset builds upon existing human annotations to examine respective
properties. Because our plausibility evaluation focuses on specific lexical agreements, we propose to evaluate explanations of probe predictions instead of word predictions in order to minimize noises introduced by the task setup.

Under this evaluation framework, we conduct a case study on some of the most widely adopted gradient-based post-hoc model explanation method on state-of-the-art neural language model architectures. Our study shows that a model explanation can either fail due to a lack of plausibility or faithfulness, and the explanations are trustworthy only when they do well with both tests. We also noticed that the performance of model explanations are generally sensitive to even minor model configuration changes. Hence, trends of explanation quality on a specific model configuration should not be over-generalized to other configurations.

We want the community to be aware that gradient-based post-hoc model explanation methods still do not generate trustworthy explanations all the time. Hence, we recommend that adopting any model explanation method as a source of knowledge about NLP models’ reasoning process should only happen after similar quantitative checks as presented in this paper are performed. We also hope our proposed test paradigm and accompanied test sets provide useful guidance to future work on evaluations of explanation methods. Our evaluation dataset and code to reproduce the analysis are available at https://github.com/shuoyangd/tarsius.
Chapter 4

Comparison of Explanations from Different Methods – Case Study on Neural Machine Translation

4.1 Introduction

In the previous chapter, we have established an evaluation framework for evaluating model explanation methods. We have also conducted a case study for gradient-based post-hoc explanation methods on language models, a category of sequence models without input context. In this chapter, we extend this study to a wider category of methods on a specific
CHAPTER 4. COMPARISON OF EXPLANATIONS FROM DIFFERENT METHODS – CASE STUDY ON NEURAL MACHINE TRANSLATION

category of sequence models with input context – the neural machine translation (NMT) models. As the readers shall see later in this section, NMT is an especially interesting case to study from a model explanation perspective, because the existing literature has provided us a platform that enables us to make quantitative comparison for an even more diverse set of model explanation methods.

Recall that in Chapter 2, we have defined the model explanation task for a machine translation model as establishing an importance distribution over the tokens in the input sequence \( x = x_0 x_1 \ldots x_I \) for a particular predicted translation token \( y_j \). Historically, this is a well-studied problem for statistical machine translation (SMT) models before the inception of NMT models, because the translation generation process for SMT would involve applying discrete phrase translation rules on the input sentence, which in turn was extracted from word-aligned parallel corpus. Because of the similarity in the output format to our model explanation task, these statistical word aligners (Brown et al., 1993; Koehn, Och, and Marcu, 2003, *inter alia*) that are used to align word translation pairs in a parallel corpus can be treated as a de-facto post-hoc model explanation method based on a dedicated explanation model.

On the other hand, one of the key innovations that led to the paradigm shift from SMT to NMT is the introduction of the encoder-decoder attention mechanism (Bahdanau, Cho, and Bengio, 2014; Luong, Pham, and Manning, 2015b), which was designed to jointly learn the
word alignment and translation tasks and could serve as an intrinsic explanation method. As shown in Equation 2.11, the encoder-decoder attention computes an attention weight \( a_{ij} \), normalized over the input tokens for a specific time step \( j \), which fits perfectly with our task definition. In fact, despite some known problems with the interpretability of the attention weights that we have reviewed in Chapter 2, early studies like Tu et al. (2016) and Tang, Sennrich, and Nivre (2018) use the attention weights between the encoder and decoder as a
way to explain and analyze the behavior of a NMT model.

Our main contribution is to employ a gradient-based post-hoc model explanation method for this task in order to make up some shortcomings with the other two methods. We conduct a quantitative evaluation for all the methods based on existing hand-crafted word alignment annotations. Our experiments show that gradient-based post-hoc methods are able to reduce Alignment Error Rate (AER) by 10-20 points over the attention weight baseline under two evaluation settings we adopt (see Figure 4.1 for an example), and beat fast-align (Dyer, Chahuneau, and Smith, 2013b) by as much as 8.7 points. We also show that architectures such as convolutional sequence-to-sequence models (Gehring et al., 2017) can implicitly learn highly interpretable word alignments, which sheds light on how future improvement should be made on these architectures.

4.2 Methods

4.2.1 Attention

As we have mentioned in Chapter 2, we compute encoder-decoder attention $a_{ij}$ in order to construct a context vector $c_j$ from the contextualized word representations built by the encoder. This context vector $c_j$ is then sent to the decoder to compute the output distribution $p(y_j \mid s_j, x)$, which is then used to predict the discrete output word $y_j^*$. When we use
attention weights to construct model explanations in the form of word alignments, we are defining the importance of a source token \( x_i \) for the prediction of \( y_j^* \) as:

\[
\psi(x_i) = a_{ij}
\] (4.1)

Note that here and for the rest of the chapter, we will use the term “attention” to refer to the encoder-decoder attention. Specifically, they do not refer to the weight of self-attention modules that only exist in the Transformer architecture, and do not establish alignment between the source and target words.

It should be noted that when beam search is used for sequence generation, the discrete prediction \( y_j^* \) might not be \( \arg\max_{y_j} p(y_j \mid s_j, x) \), as the locally optimal option may be pruned during beam search and/or not end up being a part of the final best translation hypothesis. Here, we can see an important conceptual problem regarding interpreting attention weights as an explanation for a certain prediction \( y_j^* \). Suppose for the same source sentence, there are two alternative translation hypotheses in the beam that diverge at target time step \( j \), generating \( y_j^* \) and \( y_j' \) which respectively correspond to different source words. Presumably, the source word that is aligned to \( y_j^* \) and \( y_j' \) should change correspondingly. However, this is not possible with the attention weight explanation, because the attention weight is computed \textit{before} discrete predictions \( y_j^* \) or \( y_j' \) are generated. With that, we argue that an ideal explanation algorithm should be able to adapt the explanation with a specific
output word, regardless of whether it is the locally most likely word predicted by the model.

4.2.2 Word Alignment Models

The statistical word aligners are still one of the most popular methods to induce word alignments from a parallel sentence pair, thanks to the highly accurate alignment tools like GIZA++ (Brown et al., 1993; Och and Ney, 2003) and fast-align (Dyer, Chahuneau, and Smith, 2013a). At a very high-level, given a sentence-aligned parallel corpus \( \{X, Y\} \) as input, these statistical word alignment models estimate \( p(a_{ij} \mid x_i, y_j) \) for all word pairs \( (x_i, y_j) \) within the same sentence pair \( (x, y) \), in an unsupervised manner. Note that because the alignments are induced from discrete sentences, they are not prone to the aforementioned problem that attention weights suffer from.

When we apply statistical word aligners for our model explanation task, we define the importance distribution \( \psi \) as follows:

\[
\psi(x_i) = \frac{p(a_{ij} \mid x_i, y_j)}{\sum_{i=0}^{I} p(a_{ij} \mid x_i, y_j)} \tag{4.2}
\]

Despite their high accuracy, there are issues that make statistical word aligners potentially unfit for certain cases of model auditing. The first case is for analyzing the behaviors of NMT models. Because statistical word aligners estimate a completely different set of parameters
than NMT models, their importance attribution over the source tokens may not faithfully reflect that of the NMT models. The second case is when alignments are needed before the full sentence is translated, such as the scenario in certain constrained decoding algorithms (Hasler et al., 2018) and in computer-aided translation (Bouma and Parmentier, 2014; Arcan et al., 2014). Statistical word aligners generally assume that the full sentence pair \((x, y)\) is given and exploit concepts like fertility and re-ordering to improve alignment quality, which are hard to apply for partial translations. Besides, because these methods are unsupervised, their parameters are updated on the inference input to produce optimal output. The delay required by these parameter updates themselves are also detrimental for online applications. For these cases, the current common practice is to simply fall back to the word alignments generated from attention weights.

In recent years, there were also efforts on moving beyond discrete words used by statistical word aligners and incorporating distributed representations to improve the state-of-the-art. For example, Zenkel, Wuebker, and DeNero (2019) proposed to produce word alignments by adding a dedicated alignment layer to a NMT model and finetune it on parallel corpus. Garg et al. (2019) proposed to finetune one alignment head in the Transformer model with the word alignment output from GIZA++. Stengel-Eskin et al. (2019) proposed a neural discriminative classifier that predicts whether each alignment point is aligned. Marchisio, Xiong, and Koehn (2021) proposed an enhancement to GIZA++ that takes advantage of the
CHAPTER 4. COMPARISON OF EXPLANATIONS FROM DIFFERENT METHODS – CASE STUDY ON NEURAL MACHINE TRANSLATION

embedding space geometry of the source and target language.

4.2.3 Gradient-based Post-hoc Explanations

We adopt saliency and SmoothGrad methods which we have already studied in Chapter 3 on language models. As for composition schemes, we will also focus on comparison between Vector Norm (Li et al., 2016, VN) and the Gradient · Input (GI) scheme we introduced in Section 3.4.

4.3 Experiments

4.3.1 Evaluation Method

To put our results into the context of machine translation literature, we will largely follow the existing evaluation setup for the word alignment task, which compares the discrete alignment points with human annotations and measure Alignment Error Rate (AER):

\[
\text{AER}(A, S, P) = 1 - \frac{|P \cap A| + |S \cap A|}{|S| + |A|}
\]

(4.3)

where \( A \) is the set of alignment points predicted by the system. \( S \) and \( P \) are the set of sure and possible alignment points annotated by the humans.
CHAPTER 4. COMPARISON OF EXPLANATIONS FROM DIFFERENT METHODS – CASE STUDY ON NEURAL MACHINE TRANSLATION

From the perspective of explanation quality, this evaluation is examining the plausibility of the explanation for humans. However, unlike the case with language models, where the evaluation had to be focused on a specific set of prediction where human annotations (for lexical agreements) exist, we have a human-annotated alignment for each predicted output word that also takes ambiguity into account. Hence, we will follow the practice in previous work and mostly focus on plausibility results. As a sanity check, we conduct a faithfulness test in the analysis to look at the consistency of the word alignments produced from models trained from several randomly initialized training runs.

Ideally, we would compare predicted word alignments against manually labeled word alignments between source sentences and output sentences from respective NMT models, but this would require human annotations being produced for translation outputs from each model. To mitigate the fact that we only have human annotation on the reference translations from the dataset, we instead conduct two automatic evaluations for our proposed method using resources available:

- **force decoding**: take the dataset with human-annotated alignments, run NMT models to force-decode the target side of the dataset and measure AER against the human alignment;

- **free decoding**: take the NMT prediction, obtain synthetic reference alignments between the prediction and the source and measure AER against this reference.
CHAPTER 4. COMPARISON OF EXPLANATIONS FROM DIFFERENT METHODS – CASE STUDY ON NEURAL MACHINE TRANSLATION

Notice that both automatic evaluation methods have their respective limitation: the force decoding method may force the model to predict something it deems unlikely, and thus generating noisy alignment; whereas the free decoding method lacks human-annotated references.

4.3.2 Setup

We follow the data setup in Zenkel, Wuebker, and DeNero (2019) and use the accompanied scripts of that paper\(^1\) for preprocessing. Their training data consists of 1.9M, 1.1M and 0.4M sentence pairs for German-English (de-en), English-French (en-fr) and Romanian-English (ro-en) language pairs, respectively, whereas the manually-aligned test data contains 508, 447 and 248 sentence pairs for each language pair. There is no development data provided in their setup, and it is not clear what they used for NMT system training, so we set aside the last 1,000 sentences of the training data for each language as the development set.

For free decoding experiments, we need to construct our own reference alignments for the NMT outputs. We construct such reference with the fast-align word aligner. Our reference alignment construction process is as follows: we first run automatic alignment on both sides, and take the intersection of the two outputs as “sure” alignments and the rest as “possible” alignments.

\(^1\)https://github.com/lilt/alignment-scripts
For our NMT systems, we use fairseq\textsuperscript{2} to train attention-based RNN systems (\textbf{LSTM}) (Bahdanau, Cho, and Bengio, 2014), convolution systems (\textbf{FConv}) (Gehring et al., 2017), and Transformer systems (\textbf{Transformer}) (Vaswani et al., 2017). We use the pre-configured model architectures for IWSLT German-English experiments\textsuperscript{3} to build all NMT systems. Our comparison cover the following explanation methods:

- \textit{Attention}: directly take the attention weights as soft alignment scores. For transformer, we follow the implementation in fairseq and used the attention weights from the final layer averaged across all heads;

- \textit{Smoothed Attention}: obtain multiple version of attention weights with the same data augmentation procedure as SmoothGrad and average them. This is to prove that smoothing itself does not improve the explanation quality, and has to be used together with effective explanation method;

- \textit{Gradient Saliency method} (G): applied with both VN and Gradient $\cdot$ Input (GI) composition schemes;

- \textit{SmoothGrad} (SG): applied with both VN and Gradient $\cdot$ Input (GI) composition schemes;

- \textit{fast-align}: a fast statistical word aligner based on a re-parametrized IBM Model 2

\textsuperscript{2}https://github.com/pytorch/fairseq
\textsuperscript{3}The exact model options we used are respectively fconv_iwslt_de_en, lstm_wiseman_iwslt_de_en, transformer_iwslt_de_en.
CHAPTER 4. COMPARISON OF EXPLANATIONS FROM DIFFERENT METHODS – CASE STUDY ON NEURAL MACHINE TRANSLATION

(Dyer, Chahuneau, and Smith, 2013a);

- **GIZA++**: a widely-adopted statistical word aligner implementation that contains IBM Model 1-5 as well as HMM word alignment model (Och and Ney, 2003);

- **Zenkel, Wuebker, and DeNero (2019)**: we use the best reported results from ADD+SGD method for comparison.

As mentioned before, most statistical word aligners require updating the model parameters with the full sentences in order to obtain the optimal word alignment, a constraint that is not present for other explanation methods, including the finetuning-based neural word alignment model from Zenkel, Wuebker, and DeNero (2019). To give a better picture of the performance gap with or without the parameter updates, apart from the usual offline ways to produce word alignment with statistical word aligners, we also run fast-align under the online alignment scenario, where we first train a fast-align model and decode on the test set (without updating the word aligner parameters). This is a real-world scenario in applications such as computer-aided translation (Bouma and Parmentier, 2014; Arcan et al., 2014), where we cannot practically update alignment models on-the-fly. On the other hand, we believe this is a slightly fairer comparison for methods with online alignment capabilities such as Zenkel, Wuebker, and DeNero (2019) and this work.

The data used in Zenkel, Wuebker, and DeNero (2019) did not provide a manually-aligned development set, so we tune the SmoothGrad hyperparameters (noise standard
deviation $\sigma$ and sample size $n$) on a 30-sentence subset of the German-English test data with the Transformer model. We ended up using the recommended $\sigma = 0.15 \cdot (\max(E_x) - \min(E_x))$ in the original paper$^4$ and a slightly smaller sample size $n = 30$ for speed. This hyperparameter setting is applied to the other SmoothGrad experiments as-is. For comparison with previous work, we do not exclude these sentences from the reported results, we instead mark the numbers affected to raise caution.

For all the model explanation methods based on NMT models, we follow the same procedure in Zenkel, Wuebker, and DeNero (2019) to convert soft alignment scores to hard alignment. For force decoding experiments, we also report results on symmetrized alignments, which means running word alignments on both the forward and the backward translation directions and ensembling them with heuristics to form a cleaner version of alignments. We perform the symmetrization with grow-diag-final heuristic.

### 4.3.3 Force Decoding Results

Table 4.1 shows the AER results under the force decoding setting. First, note that after applying gradient-based saliency method with GI composition scheme, AER is only reduced for FConv model but instead increases for LSTM and Transformer. The largest increase is observed for Transformer, where the AER increases by about 20 points on average. However,

$^4E_x = [e_{x0}^0, e_{x1}^0, \ldots, e_{xT}^T]$, or, equivalently, the queried embedding matrix corresponding to the source sentence $x$. 

111
## Table 4.1: Alignment Error Rate (AER) with different explanation methods, under force decoding setting. GIZA++ and fast-align Offline results are quoted from Zenkel, Wuebker, and DeNero (2019), whereas fast-align Online stands for our online alignment result (c.f. Section 4.3.2). bidir refers to the symmetrized alignment results. Best results for each architecture are marked with underlines, and best explanation/alignment results are respectively marked with boldface. Numbers affected by hyper-parameter tuning are marked with *.

<table>
<thead>
<tr>
<th>Method</th>
<th>de-en</th>
<th>en-de</th>
<th>bidir</th>
<th>fr-en</th>
<th>fr-fr</th>
<th>bidir</th>
<th>ro-en</th>
<th>en-ro</th>
<th>bidir</th>
</tr>
</thead>
<tbody>
<tr>
<td>FConv Attention</td>
<td>38.5</td>
<td>40.1</td>
<td>37.5</td>
<td>23.8</td>
<td>27.4</td>
<td>22.0</td>
<td>40.9</td>
<td>38.6</td>
<td>39.1</td>
</tr>
<tr>
<td>Smoothed Attention</td>
<td>40.2</td>
<td>43.9</td>
<td>41.2</td>
<td>24.1</td>
<td>27.4</td>
<td>22.5</td>
<td>41.5</td>
<td>39.6</td>
<td>40.4</td>
</tr>
<tr>
<td>G+VN</td>
<td>39.0</td>
<td>39.6</td>
<td>35.3</td>
<td>26.8</td>
<td>29.2</td>
<td>21.1</td>
<td>41.9</td>
<td>42.1</td>
<td>38.6</td>
</tr>
<tr>
<td>SG+VN</td>
<td>40.7</td>
<td>44.5</td>
<td>39.3</td>
<td>27.3</td>
<td>28.1</td>
<td>21.6</td>
<td>43.5</td>
<td>43.5</td>
<td>40.0</td>
</tr>
<tr>
<td>G+GI</td>
<td>33.1</td>
<td>40.5</td>
<td>26.8</td>
<td>25.2</td>
<td>22.7</td>
<td>11.9</td>
<td>37.1</td>
<td>39.4</td>
<td>29.8</td>
</tr>
<tr>
<td>SG+GI</td>
<td><strong>27.3</strong></td>
<td><strong>33.0</strong></td>
<td><strong>22.3</strong></td>
<td><strong>21.2</strong></td>
<td><strong>18.1</strong></td>
<td><strong>8.5</strong></td>
<td><strong>32.4</strong></td>
<td><strong>34.2</strong></td>
<td><strong>27.2</strong></td>
</tr>
<tr>
<td>LSTM Attention</td>
<td>42.8</td>
<td>47.5</td>
<td>36.9</td>
<td>33.7</td>
<td>38.0</td>
<td>25.8</td>
<td>47.1</td>
<td>47.0</td>
<td>40.9</td>
</tr>
<tr>
<td>Smoothed Attention</td>
<td>47.3</td>
<td>50.7</td>
<td>40.0</td>
<td>35.4</td>
<td>40.2</td>
<td>27.5</td>
<td>50.7</td>
<td>50.2</td>
<td>43.5</td>
</tr>
<tr>
<td>G+VN</td>
<td>41.0</td>
<td>43.9</td>
<td>33.5</td>
<td>32.9</td>
<td>37.1</td>
<td>23.5</td>
<td>44.5</td>
<td>44.9</td>
<td>37.5</td>
</tr>
<tr>
<td>SG+VN</td>
<td>39.4</td>
<td>43.1</td>
<td>31.5</td>
<td>32.2</td>
<td>36.2</td>
<td>22.0</td>
<td>45.7</td>
<td>46.8</td>
<td>37.7</td>
</tr>
<tr>
<td>G+GI</td>
<td>47.5</td>
<td>50.2</td>
<td>38.6</td>
<td>41.1</td>
<td>41.6</td>
<td>30.4</td>
<td>54.2</td>
<td>55.8</td>
<td>42.8</td>
</tr>
<tr>
<td>SG+GI</td>
<td>31.4</td>
<td>36.8</td>
<td>23.7</td>
<td>27.2</td>
<td>25.0</td>
<td>13.8</td>
<td>40.4</td>
<td>39.9</td>
<td>32.0</td>
</tr>
<tr>
<td>Transformer Attention</td>
<td>53.4</td>
<td>58.6</td>
<td>42.3</td>
<td>48.1</td>
<td>48.7</td>
<td>33.8</td>
<td>51.6</td>
<td>51.1</td>
<td>43.3</td>
</tr>
<tr>
<td>Smoothed Attention</td>
<td>55.8</td>
<td>56.1</td>
<td>48.6</td>
<td>42.5</td>
<td>47.5</td>
<td>32.9</td>
<td>57.5</td>
<td>57.6</td>
<td>51.5</td>
</tr>
<tr>
<td>G+VN</td>
<td>51.1</td>
<td>56.2</td>
<td>43.7</td>
<td>43.6</td>
<td>47.9</td>
<td>39.9</td>
<td>46.7</td>
<td>48.4</td>
<td>35.5</td>
</tr>
<tr>
<td>SG+VN</td>
<td>36.4</td>
<td>45.8</td>
<td>30.3</td>
<td>27.0</td>
<td>25.5</td>
<td>15.6</td>
<td>41.3</td>
<td>39.9</td>
<td>33.7</td>
</tr>
<tr>
<td>G+GI</td>
<td><strong>77.7</strong></td>
<td><strong>78.2</strong></td>
<td><strong>77.4</strong></td>
<td><strong>69.1</strong></td>
<td><strong>72.5</strong></td>
<td><strong>74.5</strong></td>
<td><strong>74.6</strong></td>
<td><strong>75.2</strong></td>
<td><strong>71.0</strong></td>
</tr>
<tr>
<td>SG+GI</td>
<td>*36.4</td>
<td>43.0</td>
<td>*39.0</td>
<td>29.7</td>
<td>25.9</td>
<td>13.3</td>
<td>41.2</td>
<td>41.4</td>
<td>32.7</td>
</tr>
<tr>
<td>fast-align Offline</td>
<td>28.4</td>
<td>32.0</td>
<td>27.0</td>
<td>16.4</td>
<td>15.9</td>
<td>10.5</td>
<td>33.8</td>
<td>35.5</td>
<td>32.1</td>
</tr>
<tr>
<td>fast-align Online</td>
<td>30.8</td>
<td>34.4</td>
<td>30.0</td>
<td>18.8</td>
<td>16.8</td>
<td>13.6</td>
<td>37.1</td>
<td>41.1</td>
<td>35.9</td>
</tr>
<tr>
<td><em>Zenkel, Wuebker, and DeNero, 2019</em></td>
<td><strong>26.6</strong></td>
<td><strong>30.4</strong></td>
<td><strong>21.2</strong></td>
<td><strong>23.8</strong></td>
<td><strong>20.5</strong></td>
<td><strong>10.0</strong></td>
<td><strong>32.3</strong></td>
<td><strong>34.8</strong></td>
<td><strong>27.6</strong></td>
</tr>
<tr>
<td>GIZA++</td>
<td><strong>21.0</strong></td>
<td><strong>23.1</strong></td>
<td><strong>21.4</strong></td>
<td><strong>8.0</strong></td>
<td><strong>9.8</strong></td>
<td><strong>5.9</strong></td>
<td><strong>28.7</strong></td>
<td><strong>32.2</strong></td>
<td><strong>27.9</strong></td>
</tr>
</tbody>
</table>
when SG+GI is applied, we observe a sharp drop in AER, which ends up with 10-20 points lower than the attention weight baseline. We can also see that this is not just an effect introduced by input noise, as the same smoothing procedure for attention increases the AER most of the times. To summarize, at least under force decoding settings, SG+GI obtains word alignment explanations of much higher quality than the attention weight baseline.

As for VN composition scheme, for FConv and LSTM architectures, it is not only consistently worse than results from GI, but at times also worse than attention. Besides, the effect of SmoothGrad is also not as consistent on VN as GI. Although with the Transformer model, the VN composition scheme obtained better AER than GI under several settings, it is still pretty clear overall that the superior mathematical soundness of GI is translated into better explanation quality. This corroborates our analysis results on language models in section 3.5.3.

While the GIZA++ model obtains the best alignment result in Table 4.1, most of SG+GI explanations we obtained from FConv surpasses the alignment quality of fast-align (either Online or Offline), sometimes by as much as 8.7 points (symmetrized ro<en result). SG+GI are also largely on-par with (Zenkel, Wuebker, and DeNero, 2019). These are notable results as gradient-based post-hoc explanation methods has no extra parameter updated to optimize the quality of alignment. On the other hand, this also indicates that it is possible to induce

---

5While Ghader and Monz (2017) showed that the AER obtained by LSTM model is close to that of GIZA++, our experiments yield a much larger difference. We think this is largely due to the fact that we choose to train our model with BPE, while Ghader and Monz (2017) explicitly avoided doing so.
high-quality alignments from NMT model without modifying its parameters, showing that it has acquired such information in an implicit way. Most interestingly, although NMT is often deemed as performing poorly under low-resource setting, the explanation seems to work relatively well on ro<en language pair, which happens to be the language pair that we have least training data for. We think this is a phenomenon that merits further exploration.

Besides, it can be seen that for all reported methods, the overall order for the number of alignment errors is FConv < LSTM < Transformer. To the best of our knowledge, this is also a novel observation, as no one has analyzed attention weights of FConv with other architectures before. We can also observe that while gradient-based explanation is not strong enough to fully bridge the gap of the attention noise level between different model architecture, it does manage to narrow the difference in some cases.

4.3.4 Free Decoding Results

Table 4.2 shows the result under the free decoding setting. The trend in this group of experiment is similar to Table 4.1, except that Transformer occasionally outperforms LSTM. We think this is mainly due to the fact that Transformer generates higher quality translations, but could also be partially attributed to the noise in fast-align reference. Also, notice that the AER numbers are also generally lower compared to Table 4.1 under this setting. One reason is that our model is aligning output with which it is most confident, so less noise
### Table 4.2: Alignment Error Rate (AER) with different explanation models, under free decoding setting. See the caption of Table 4.1 for notations.

<table>
<thead>
<tr>
<th>Method</th>
<th>de-en</th>
<th>en-de</th>
<th>en-fr</th>
<th>fr-en</th>
<th>ro-en</th>
<th>en-ro</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCConv</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attention</td>
<td>27.4</td>
<td>24.2</td>
<td>20.7</td>
<td>23.6</td>
<td>32.5</td>
<td>25.6</td>
</tr>
<tr>
<td>Smoothed Attention</td>
<td>29.4</td>
<td>29.0</td>
<td>21.1</td>
<td>23.6</td>
<td>33.7</td>
<td>26.7</td>
</tr>
<tr>
<td>G+VN</td>
<td>29.3</td>
<td>23.5</td>
<td>25.0</td>
<td>23.7</td>
<td>33.9</td>
<td>27.9</td>
</tr>
<tr>
<td>SG+VN</td>
<td>31.2</td>
<td>30.4</td>
<td>24.1</td>
<td>24.0</td>
<td>35.6</td>
<td>30.1</td>
</tr>
<tr>
<td>G+GI</td>
<td>18.2</td>
<td>20.0</td>
<td>20.2</td>
<td>14.3</td>
<td>24.9</td>
<td>22.8</td>
</tr>
<tr>
<td>SG+GI</td>
<td><strong>13.7</strong></td>
<td><strong>14.2</strong></td>
<td><strong>17.0</strong></td>
<td><strong>10.6</strong></td>
<td><strong>21.4</strong></td>
<td><strong>17.4</strong></td>
</tr>
<tr>
<td>LSTM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attention</td>
<td>33.6</td>
<td>34.6</td>
<td>32.5</td>
<td>32.3</td>
<td>36.5</td>
<td>31.7</td>
</tr>
<tr>
<td>Smoothed Attention</td>
<td>38.2</td>
<td>39.5</td>
<td>34.3</td>
<td>35.2</td>
<td>44.2</td>
<td>36.3</td>
</tr>
<tr>
<td>G+VN</td>
<td>34.1</td>
<td>32.5</td>
<td>33.6</td>
<td>33.7</td>
<td>36.6</td>
<td>32.1</td>
</tr>
<tr>
<td>SG+VN</td>
<td>30.8</td>
<td>29.4</td>
<td>31.8</td>
<td>32.1</td>
<td>38.9</td>
<td>34.8</td>
</tr>
<tr>
<td>G+GI</td>
<td>35.9</td>
<td>36.7</td>
<td>40.2</td>
<td>36.3</td>
<td>44.1</td>
<td>43.1</td>
</tr>
<tr>
<td>SG+GI</td>
<td>20.5</td>
<td>21.9</td>
<td>26.0</td>
<td>19.1</td>
<td>32.6</td>
<td>27.5</td>
</tr>
<tr>
<td>Transformer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attention</td>
<td>50.2</td>
<td>53.0</td>
<td>50.4</td>
<td>48.5</td>
<td>44.9</td>
<td>41.9</td>
</tr>
<tr>
<td>Smoothed Attention</td>
<td>51.4</td>
<td>49.0</td>
<td>44.5</td>
<td>47.3</td>
<td>49.9</td>
<td>48.9</td>
</tr>
<tr>
<td>G+VN</td>
<td>49.9</td>
<td>51.2</td>
<td>49.4</td>
<td>51.5</td>
<td>42.9</td>
<td>40.8</td>
</tr>
<tr>
<td>SG+VN</td>
<td>27.8</td>
<td>35.3</td>
<td>28.3</td>
<td>22.3</td>
<td>30.5</td>
<td>26.5</td>
</tr>
<tr>
<td>G+GI</td>
<td>76.7</td>
<td>76.6</td>
<td>77.1</td>
<td>78.9</td>
<td>71.9</td>
<td>74.0</td>
</tr>
<tr>
<td>SG+GI</td>
<td><em>26.6</em></td>
<td><em>31.0</em></td>
<td><em>30.0</em></td>
<td><em>21.4</em></td>
<td><em>30.0</em></td>
<td><em>28.2</em></td>
</tr>
</tbody>
</table>
CHAPTER 4. COMPARISON OF EXPLANATIONS FROM DIFFERENT METHODS – CASE STUDY ON NEURAL MACHINE TRANSLATION

Figure 4.2: SmoothGrad explanation of FConv de-en Model with VN and GI composition schemes.

should be expected in the model behavior. On the other hand, by qualitatively comparing the reference translation in the test set and the NMT output, we find that it is generally easier to align the translation as it is often a more literal translation.

4.4 Analysis

4.4.1 Comparison with Vector Norm Composition Scheme

Recall in Section 3.4 and 3.5.3 we mentioned that the Vector Norm composition scheme is not a good fit for the language model explanation task due to the lack of polarity. In this section, we will see that it is also the case for the explanation of NMT model predictions. Figure 4.2 shows a case where this problem occurs in our German-English experiments.
CHAPTER 4. COMPARISON OF EXPLANATIONS FROM DIFFERENT METHODS – CASE STUDY ON NEURAL MACHINE TRANSLATION

Note that in Subfigure (a), the source word *nur* has high importance on several target words, e.g. *should*, but the word *nur* is actually not translated in the reference. On the other hand, as shown in Subfigure (b), our method correctly assigns negative (shown as white) or small positive values at all time steps for this source word. Specifically, the importance score of *nur* for *should* is negative with large magnitude, indicating significant negative contributions to the prediction of that target word. Hence, a good word alignment explanation should strongly avoid aligning them.

### 4.4.2 SmoothGrad

Tables 4.1 and 4.2 show that SmoothGrad is a crucial factor to reduce AER, especially for Transformer. Figure 4.3 shows the explanation of the same German-English sentence pair by SG+GI, but with Transformer and different SmoothGrad noise levels. Specifically, Subfigures (a) and (c) corresponds to our Grad and SmoothGrad experiments in Table 4.1. By comparing Subfigures (a) and (c), we notice that (1) G+VN explanations obtained from the Transformer model are extremely noisy, and (2) the explanations of SmoothGrad are not only a smoother version of the naïve gradient output, but also gains new information by performing extra forward and backward evaluations with the noisy input. For example, compare the alignment point between source word *wir* and target word *we*: in Subfigure (a), this word pair has very low importance score, but in (c), they become the most likely
CHAPTER 4. COMPARISON OF EXPLANATIONS FROM DIFFERENT METHODS – CASE STUDY ON NEURAL MACHINE TRANSLATION

Figure 4.3: Explanation of Transformer de-en model with different SmoothGrad noise values \( \sigma \) (\( n = 30 \)).
alignment pair for that target word.

Referring back to the motivation for using SmoothGrad in Equation 2.14, we think the observations above verify that the Transformer model is a case where very high non-linearities occur almost everywhere in the parameter space, such that the feature importance score obtained from local perturbation is a very poor representation of the global importance almost all the time. On the other hand, this is also why the Transformer especially relies on SmoothGrad to work well, as the perturbation will give a better estimation of the global importance.

It could also be observed from Subfigures (b) and (d) that when the noise is too moderate, the evaluation does not deviate enough from the original spot to gain non-local information, and in (d) it deviates too much and hence the resulting alignment is almost random. Intuitively, the noise parameter $\sigma$ should be sensitive to the model architecture or even specific input feature values, but interestingly we end up finding that a single choice from the computer vision literature works well with all of our systems. We encourage future work to conduct more comprehensive analysis of the effect of SmoothGrad on more complicated architectures beyond convolutional neural nets.
CHAPTER 4. COMPARISON OF EXPLANATIONS FROM DIFFERENT METHODS – CASE STUDY ON NEURAL MACHINE TRANSLATION

4.4.3 Alignment Dispersion

We run German-English alignments under several different SmoothGrad noise deviation $\sigma$ and report their dispersion as measured by entropy of the (soft) alignment distribution averaged by number of target words. Results are summarized in Table 4.3, where lower entropy indicates more peaky alignments. First, we observe that dispersion of importance distribution gets higher as we increase $\sigma$, which aligns with the observations in Figure 4.3. It should also be noted that the alignment dispersion is consistently lower for free decoding than force decoding. This verifies our conjecture that the force decoding setting might introduce more noise in the model behavior, but judging from this result, that gap seems to be minimal. Comparing different architectures, the dispersion of attention weights does not correlate well with the dispersion of importance distribution. We also notice that, while the Transformer attention explanation consistently results in higher AER, its dispersion is lower than the other architectures, indicating that with attention, a lot of the probability

<table>
<thead>
<tr>
<th></th>
<th>attention $\sigma = 0$</th>
<th>$\frac{\sigma_{\text{max}} - \sigma_{\text{min}}}{\sigma_{\text{max}} - \sigma_{\text{min}}}$ = 0.05</th>
<th>$\frac{\sigma_{\text{max}} - \sigma_{\text{min}}}{\sigma_{\text{max}} - \sigma_{\text{min}}}$ = 0.15</th>
<th>$\frac{\sigma_{\text{max}} - \sigma_{\text{min}}}{\sigma_{\text{max}} - \sigma_{\text{min}}}$ = 0.3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FConv</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>force</td>
<td>2.09</td>
<td>1.36</td>
<td>1.48</td>
<td>1.89</td>
</tr>
<tr>
<td>free</td>
<td>2.00</td>
<td>1.34</td>
<td>1.43</td>
<td>1.79</td>
</tr>
<tr>
<td><strong>LSTM</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>force</td>
<td>1.75</td>
<td>1.63</td>
<td>2.02</td>
<td>2.54</td>
</tr>
<tr>
<td>free</td>
<td>1.65</td>
<td>1.57</td>
<td>1.91</td>
<td>2.46</td>
</tr>
<tr>
<td><strong>Transformer</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>force</td>
<td>1.73</td>
<td>1.91</td>
<td>2.63</td>
<td>2.76</td>
</tr>
<tr>
<td>free</td>
<td>1.69</td>
<td>1.89</td>
<td>2.62</td>
<td>2.74</td>
</tr>
</tbody>
</table>

Table 4.3: Alignment distribution entropy for selected de-en models.
mass might be concentrated in the wrong place more often. This corroborates the finding in Raganato and Tiedemann (2018).

### 4.4.4 Stability Analysis

We run de-en, en-fr, ro-en models multiple times to evaluate the variance of word alignment explanation and its relationship with the choice of explanation method. Results are shown in Table 4.4. In general we find the variance of FConv and LSTM comparable, and that of Transformer is significantly higher. Unfortunately, none of the model explanation methods explored here reduces the variance of explanation.

### 4.5 Discussion

For the attention and gradient-based explanation methods, there are three different sources of word alignment errors: the limitation of NMT models on learning word alignments, the limitation of model explanation method on recovering interpretable word alignments from the model, and the annotation variations caused by the intrinsic ambiguity of the word alignment task. Our evaluation controls the first factor by having different methods to explain the same model, hence enabling our comparison across different explanation methods. However, from a model analysis perspective, it is still very hard to disentangle the
first two factors given the current state of model interpretability research. A possible path is to design a probing task to quantify the first source of error, thus separating the two sources of errors. It should also be noted that on the other hand, only the last two sources of errors exists for the statistical word aligners.

Besides, in this paper, we only explored two explanation methods among many others available (Montavon, Samek, and Müller, 2018). In our preliminary study, we also experimented with guided back-propagation (Springenberg et al., 2014), a frequently used explanation method in computer vision, which did not work well for our problem. We
suspect that there is a gap between applying these methods on mostly-convolutional architectures in computer vision and architectures with more non-linearities in NLP. We hope the future research from the NLP and machine learning communities could bridge this gap.

Finally, as mentioned before, we are only conducting approximate evaluation to measure the ability of our explanation method. An immediate future work would be evaluating this on human-annotated translation outputs generated by the NMT system.

4.6 Conclusion

We conduct a case study on NMT models to compare model explanations generated by different methods. Our comparison shows that gradient-based post-hoc explanation methods achieve good balance between generating explanations that are plausible to humans and being faithful to the model decision mechanism. When compared with the attention mechanism, post-hoc methods generate explanations that are far more plausible to humans. When compared with statistical word alignment models, post-hoc methods have exposure to the original inference model, thus being more faithful to the model’s decision mechanism. Our empirical results also probe into the NMT black-box and reveal that even without any special architecture or training algorithm, some NMT models have already implicitly learned interpretable word alignments of comparable quality to fast-align. The model and code for our experiments are available at https://github.com/shuoyangd/meerkat.
Intended to be blank.
Part II

Quality
Intended to be blank.
Chapter 5

Comparison of Quality Estimations from Different Methods

5.1 Introduction

We now turn to the task of quality estimation. As we have described in Chapter 2, the two main ways of performing the post-hoc user-centric model audit are either extracting signals from the original model or building a dedicated model that generates the signals for audit. Among these two different categories of methods, Fomicheva et al. (2020) carried out a systematic comparison, but for the segment-level quality estimation task, while we focus on the word level. Interestingly, their work showed that despite lack of supervision
CHAPTER 5. COMPARISON OF QUALITY ESTIMATIONS FROM DIFFERENT METHODS

from human judgments, some post-hoc methods that extract audit signals from the original model perform on-par with dedicated quality estimation models. In this chapter, we explore whether a similar trend holds for word-level quality estimation.

In both this and the next chapter, we will be focusing on the case of sequence generation models with an input context – specifically, on the task of machine translation. The reason for this choice is that the notion of “quality” is very hard to quantify for a sequence generated without the input context, such as the ones generated by a language model. In fact, in the existing literature, evaluations for language models are generally done by measuring the models’ capability of assigning a relatively high probability for a sample of fluent, human-generated text, measured by perplexity. The specific choice for machine translation among the other sequence generation tasks is largely driven by the availability of well-established tasks and test sets, thus enabling us to easily draw comparisons across different methods.

Our comparison in this chapter will be focusing on three different methods: (1) Translation Probability (TP), (2) Monte Carlo Dropout (Gal and Ghahramani, 2016, MCD), and (3) Predictor-Estimator Model (Kim et al., 2017, PredEst). The first two are methods based on the original model, while the last one is the current state-of-the-art of dedicated quality estimation models. For methods (1) and (2), we also consider a case where a stronger translation model (different from the original translation model used for generating the translation output) is used to score each token in the translation output, thus serving as a
CHAPTER 5. COMPARISON OF QUALITY ESTIMATIONS FROM DIFFERENT METHODS

quality estimation model. This allows us to take advantage of large-scale self-supervised training for both TP and MCD methods, as with the case for most existing PredEst models.

5.2 Methods

Recall that our problem definition for (word-level) quality estimation in Section 2.4.1 was to generate a binary word quality label for each word in the output sequence, and optionally, a binary gap quality label between every two words in the output sequence. As the reader shall see in the method description, for some of the methods we will be comparing in this chapter, it is not straightforward to generate the quality labels for gaps. Hence, for all the experiments in this chapter, we drop the optional quality labels for gaps and only focus on quality labels for words.

Translation Probability (TP) As the name suggests, the binary quality label is solely determined by the conditional probability of each word $p(y_j \mid s_j, x)$:

$$l^w_j = \begin{cases} 
OK, & \text{if } p(y_j \mid s_j, x) > \beta \\
BAD, & \text{if } p(y_j \mid s_j, x) \leq \beta 
\end{cases} \quad (5.1)$$

where $y_j$ is the $(j + 1)$-th word in the translation output, $s_j$ is the prefix of the first $j$ words $y_0y_1 \ldots y_{j-1}$ of the translation output. $\beta$ is a threshold value. We will talk about how $\beta$ is
CHAPTER 5. COMPARISON OF QUALITY ESTIMATIONS FROM DIFFERENT METHODS

selected later in this section.

Monte Carlo Dropout (MCD) Proposed by Gal and Ghahramani (2016), this method estimates prediction uncertainty by performing dropout on the neurons of each layer. The decision rule is very similar to the translation probability method above (Equation 5.1). The only difference is that \( p(y_j \mid s_j, x) \) is no longer estimated by a single forward pass of the translation model, but rather by averaging multiple forward passes with dropout applied, as shown in Equation 2.21.

Predictor-Estimator Proposed by Kim et al. (2017), this method adopts a two-stage approach that is very similar to the current “pre-train and finetune” paradigm. It first trains a predictor model, which is similar to Translation Language Model (Conneau and Lample, 2019, TLM) that predicts the probability of the target word \( y_j \) with the input sentence \( x \), the output prefix \( y_0y_1\ldots y_{j-1} \), and the output suffix \( y_{j+1}y_{j+2}\ldots y_J \) as the context\(^1\). The second stage is fine-tuning on a small task-specific dataset with an estimator model added on top of the feature extracted by the predictor model.

Our results of the predictor-estimator model are based on the widely-adopted OpenKiwi model implementation\(^2\) (Kepler et al., 2019b), which diverges from the original predictor-estimator proposal in a few aspects: (1) we no longer train our own predictor model, but rather utilize the existing pre-trained XLM-RoBERTa encoder (Conneau et al., 2020); (2)

\(^1\)Note that an usual machine translation model will not include the output suffix. Because of that, TLM is more similar to a masked language model and cannot be used for sequence generation.

\(^2\)https://github.com/Unbabel/OpenKiwi
CHAPTER 5. COMPARISON OF QUALITY ESTIMATIONS FROM DIFFERENT METHODS

for the estimator model, a simple linear layer is used instead of a Bi-LSTM; (3) because
the encoder is now multilingual, we simply concatenate the source and the target sentence
(following the format as stated in Moura et al. (2020)) instead of training separate encoders
for both source and target languages.

5.3 Evaluation

As we have mentioned in Section 2.4.2, while most existing work on quality estimation
models for machine translation adopts F1 score or Matthews Correlation Coefficient (MCC)
as the evaluation metric, there was also some recent work on the calibration of the neural
machine translation model that uses Expected Calibration Error (ECE) as the evaluation
metric (Kumar and Sarawagi, 2019; Wang et al., 2020c). Such work on model calibration
resembles with the TP and MCD methods we are comparing in this chapter. Because of
that, we would like to clarify with a toy example why we think calibration is not a good
evaluation for our task of interest.

For simplicity, let’s consider two binary classification models $\mathcal{M}_A$ and $\mathcal{M}_B$. Suppose
both $\mathcal{M}_A$ and $\mathcal{M}_B$ have an accuracy (i.e., the fraction of correct/true examples among
all examples) of 50%. The difference is that $\mathcal{M}_A$ assigns a probability of 1.0 for all of
its correct predictions and a probability of 0.51 for all of its incorrect ones. $\mathcal{M}_B$, instead,
assigns a probability of 0.51 for all predictions.
CHAPTER 5. COMPARISON OF QUALITY ESTIMATIONS FROM DIFFERENT METHODS

Now we calculate ECE for these two models according to Equation 2.25:

\[
ECE_A = 0.5 \times |1.0 - 1.0| + 0.5 \times |0.0 - 0.51| = 0.255
\]  \hspace{1cm} (5.2)
\[
ECE_B = 1.0 \times |0.5 - 0.51| = 0.01
\]  \hspace{1cm} (5.3)

Hence, we see that \( \mathcal{M}_B \) has a smaller calibration error than \( \mathcal{M}_A \), hence \( \mathcal{M}_B \) is a better-calibrated model. However, it should be self-evident that \( \mathcal{M}_A \) is actually a better fit for our task, because any decision boundary \( \beta \in (0.51, 1.0) \) would allow us to achieve a perfect identification of correctly and incorrectly classified samples, while such decision boundary does not exist for \( \mathcal{M}_B \). The caveat here is that among equally calibrated models, the quality estimation task implicitly prefers more confident predictions to ambivalent ones.

Because of this, we do not analyze calibration of the model prediction score, but rather directly evaluate the performance of the discrete quality labels. We follow the practice in the recent years of WMT quality estimation task (Specia et al., 2021) and choose MCC as the main metric.
CHAPTER 5. COMPARISON OF QUALITY ESTIMATIONS FROM DIFFERENT METHODS

5.4 Experiments

5.4.1 Configurations

Data  We use the dataset for WMT 2021 quality estimation shared task\(^3\). Each data instance comes in the form of a translation triplet, which includes the input/source segment, an output/target segment generated by an NMT model, and a segment post-edited by humans based on the NMT output. For all language pairs, there are 7000 triplets for training, 1000 for development, and 1000 for testing. Our experiments cover 6 language pairs of the dataset: Nepali-English (ne-en), Sinhala-English (si-en), Estonian-English (et-en), Romanian-English (ro-en), English-German (en-de), English-Chinese (en-zh). The selection of language pairs covers different available resource levels for the inference model training (ne-en and si-en for low-resource, et-en and ro-en for medium-resource, and en-de and en-zh for high-resource).

The dataset also includes reference quality labels that are converted from the human post edits, with the tercom toolkit\(^4\). Because of the setup of the WMT 2021 quality estimation shared task, the reference also includes quality labels for gaps. Because TP and MCD methods cannot generate quality labels for gaps, we simply remove all the quality labels for gaps and focus our evaluation on the quality labels for words.

\(^3\)https://github.com/facebookresearch/mlqe
\(^4\)https://www.cs.umd.edu/~snover/tercom/
CHAPTER 5. COMPARISON OF QUALITY ESTIMATIONS FROM DIFFERENT METHODS

Note that both TP and MCD results do not require the usage of the training set. Among the methods we compare, only the PredEst model used the training set to fine-tune the estimator module.

**Model** For TP and MCD methods, we experimented with two sets of NMT models to generate the word-level posterior probability. The first set is the original bilingual translation models used by the shared task organizers to generate the target translations in the dataset. We follow the same data preprocessing procedure to generate the word-level scores\(^5\), but use the fairseq option `--score-reference` to force decode the target segment and add MCD-related options when applicable. The second is the M2M-100 multilingual NMT model (Fan et al., 2020a), which is a translation model that takes advantage of large-scale multilingual training. In order to fit the model on our GPU, we used the smallest 418M model dump of M2M-100. For this model, we also follow their documented data preprocessing procedure\(^6\) but adopt force decoding and MCD-related options like the original inference models.

For PredEst methods, because we adopted the same setup as WMT 2021 quality estimation shared task, we simply cite word-level MCC results from the baseline model of the shared task.

**Threshold Tuning** For both TP and MCD methods, we need to select a decision threshold \( \beta \) that converts continuous probability scores into discrete labels. For our experi-

---


\(^6\) [https://github.com/pytorch/fairseq/blob/main/examples/m2m_100/README.md](https://github.com/pytorch/fairseq/blob/main/examples/m2m_100/README.md)
ment, we tune the decision threshold on the development set, where we repeatedly generate discrete labels with different thresholds and select the one with the best MCC value. We conduct this tuning process with Powell search (Powell, 1964), which is a gradient-free heuristic optimization method.

**Hyperparameters** The only hyperparameters we need to tune are the dropout rate and sample size for MCD experiments. We individually tuned the dropout rate with a grid search of \{0.1, 0.2, 0.3, 0.4, 0.5\} for the original models and M2M-100 model on en-de development set and select the one with the best word-level MCC. As a result, we use a dropout rate of 0.2 for the original models and 0.1 for the M2M-100 model for our comparison. For the sample size, We follow the recommendation in Fomicheva et al. (2020) and used a value of 10.

### 5.4.2 Results

Table 5.1 shows results. First, it is clear that the PredEst model outperforms both TP and MCD across the board, for both the original inference model and the M2M-100 model. This is different from the case of sentence-level quality estimation in Fomicheva et al. (2020), where MCD performed on-par with PredEst models based on pre-trained encoders for several different language pairs. When we compare across language pairs with different resource levels for the inference model, we see that the performance gap between
CHAPTER 5. COMPARISON OF QUALITY ESTIMATIONS FROM DIFFERENT METHODS

<table>
<thead>
<tr>
<th></th>
<th>low-resource</th>
<th>mid-resource</th>
<th>high-resource</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ne-en</td>
<td>si-en</td>
<td>et-en</td>
</tr>
<tr>
<td>PredEst</td>
<td>0.440</td>
<td>0.425</td>
<td>0.461</td>
</tr>
<tr>
<td>Orig. Model</td>
<td>0.187</td>
<td>0.180</td>
<td>0.278</td>
</tr>
<tr>
<td>MCD</td>
<td>0.187</td>
<td>0.189</td>
<td>0.324</td>
</tr>
<tr>
<td>M2M-100</td>
<td>0.187</td>
<td>0.232</td>
<td>0.321</td>
</tr>
<tr>
<td>M2M-100</td>
<td>0.238</td>
<td>0.235</td>
<td>0.320</td>
</tr>
</tbody>
</table>

Table 5.1: Comparison of Matthews Correlation Coefficient on translation words (word MCC) between dedicated Predictor-Estimator (PredEst) quality estimation model, quality estimations built from translation probability (TP), and quality estimations built from Monte-Carlo Dropout (MCD)

the dedicated model and TP/MCD is largest for low-resource language pairs, but becomes smaller as the resource level increases. This is also very different from the observation in Fomicheva et al. (2020), which showed no clear change of such comparison with different resource levels.

Nevertheless, we observe a similar trend where MCD improves quality estimation over TP for most of the cases, except for en-zh, where there was a performance degradation with MCD for both models used for scoring. But like Fomicheva et al. (2020), there is a lot of variation as to how much improvement MCD can bring over TP. On the other hand, using a stronger NMT model than the original inference model like M2M-100 is helpful for low and mid-resource language pairs, but hurts performance for higher-resource ones.
CHAPTER 5. COMPARISON OF QUALITY ESTIMATIONS FROM DIFFERENT METHODS

5.5 Conclusion

We conclude that, unlike segment-level quality estimation, extracting signals from an NMT model for quality estimation is not yet comparable to a dedicated model for quality estimation in terms of performance. Keep in mind that for the PredEst model, we are only using the baseline OpenKiwi model from the WMT 2021 quality estimation shared task. The performance gap between PredEst models and TP/MCD methods could be even larger when we also take the winning systems from the shared task into account.

There are several possible reasons that may account for this difference of trend between segment-level and word-level tasks. Note that an NMT model only has access to the left context when estimating the posterior probability of a word \( p(y_j | s_j, x) \), while bidirectional context is available and utilized by PredEst models \( p(y_j | s_j, x, y_{j+1}y_{j+2} \ldots y_J) \). While left-to-right factorization is a good way of estimating the segment-level probability, it may not be the optimal way to factorize such probability mass to each word for the purpose of quality estimation. A possible way out is to give up left-to-right factorization and explore the usage of non-autoregressive NMT models, which is the topic of the next chapter.

Beyond the choice of models, the way reference quality tags are currently generated could also be overly dependent on the post-editing style of a specific post-editor, but does not adequately take the ambiguity in the notion of “word-level quality” into account. This might have partially contributed to the reliance on the supervision from human post-editing.
CHAPTER 5. COMPARISON OF QUALITY ESTIMATIONS FROM DIFFERENT METHODS

in order to achieve optimal performance. However, this discussion on the task setup is beyond the scope of this thesis.

Because of this performance disparity, together with the inability to predict quality labels for gaps, we will focus our discussion on further improving dedicated quality estimation models for the rest of the thesis.
Chapter 6

Improving Quality Estimation Models

Through Levenshtein Training

6.1 Introduction

In this chapter, we explore whether we can improve the dedicated quality estimation models beyond the state-of-the-art set by Predictor-Estimator framework (Kim et al., 2017; Kepler et al., 2019b). Based on our previous experiments in Chapter 5, we focus our exploration on leveraging non-autoregressive NMT models to build better quality estimation models, specifically an architecture called Levenshtein Transformer (Gu, Wang, and Zhao, 2019, LevT). The Levenshtein Transformer is a good fit for the word-level quality estimation
CHAPTER 6. IMPROVING QUALITY ESTIMATION MODELS THROUGH LEVENSHTEIN TRAINING

task because it is trained to generate translations by starting with an empty output sequence and iteratively performing edits (both insertions and deletions) on the sequence. Because of this special training and decoding procedure, the model should have already learned to edit an existing translation sentence without supervision from post-edited translations. We then use multi-stage transfer learning to teach the model to perform the actual quality estimation task, first on artificially-crafted pseudo post-editing, then on real human post-edited data.

We conduct our experiments on two different datasets. On the dataset for WMT 2020 word-level quality estimation shared task (Specia et al., 2020a), we show that our method achieves better performance than the currently widely adopted Predictor-Estimator framework under the data-constrained setting, while also being competitive when compared with the high-ranking submissions in the WMT 2020 shared task under the unconstrained setting. On the devtest set for WMT 2021 word-level quality estimation shared task (Specia et al., 2021), we extended experiments to more language pairs and conducted further analysis on certain configurations of the training process. Our entry to the WMT 2021 word-level quality estimation shared task is also competitive compared to other participants.
CHAPTER 6. IMPROVING QUALITY ESTIMATION MODELS THROUGH LEVENSHEIN TRAINING

6.2 Levenshtein Training for Word-level Quality Estimation

6.2.1 Motivation

Because our task is to assess the quality of translations, it is intuitive that translation knowledge should be helpful. However, many multilingual pre-trained models (e.g. mBERT, mBART, XLM-RoBERTa) are not trained on any parallel data, nor can they to generate translations out-of-the-box. Hence, starting from an NMT model is a potentially promising approach that may lead to performance improvements.

However, as we have seen in Chapter 5, this path presents a few obstacles that have so far led to subpar performance:

1. Most NMT models are autoregressive, which means that the posterior probability of an output word $y_j$ is conditioned on the previously generated prefix $s_j = y_0y_1 \ldots y_{j-1}$.
   However, note that, different from machine translation decoding, we have a complete machine translation output as the input to the quality estimation task, which means that the contexts on both sides of $y_j$ are available.

2. It is not straightforward for most of the unsupervised quality estimation models to generate quality labels for gaps between words, which would be useful to indicate
CHAPTER 6. IMPROVING QUALITY ESTIMATION MODELS THROUGH LEVENSHTEIN TRAINING

missing words in the translation.

This has led us to explore non-autoregressive models, which diverge from the original left-to-right factorization defined in Equation 2.1. Recall that, under left-to-right factorization, at each time step, a sequence model estimates the posterior probability \( p(y_j \mid s_j, x) \), where \( s_j \) denotes the output prefix \( y_0 y_1 \ldots y_{j-1} \). At a very high-level, there are two types of non-autoregressive sequence models:

- **Fully Non-Autoregressive Model**: These models completely eliminate the dependency between each individual output words and estimate the probability \( p(y_j \mid x) \). Hence, they are able to generate all the words in the output sequence \( y \) in parallel.

- **Iterative Non-Autoregressive Model**: These models generate the output sequence iteratively by estimating the probability \( p(y_j^{(k)} \mid y^{(k-1)}, x) \). Hence, while all the words of the output sequence \( y^{(k)} \) generated at iteration \( k \) cannot be dependent on each other, they are conditioned on the output sequence from the previous iteration \( y^{(k-1)} \).

Recall that in the quality estimation task, we would like to include both the source sentence and the machine translation output into the context. For this reason, among these two types of non-autoregressive models, it is more natural to start with the iterative non-autoregressive model \( p(y_j^{(k)} \mid y^{(k-1)}, x) \) and adapt it to predict \( p(l_w^j \mid y, x) \) and \( p(l_g^j \mid y, x) \) for the quality estimation task.
CHAPTER 6. IMPROVING QUALITY ESTIMATION MODELS THROUGH LEVENSHTEIN TRAINING

6.2.2 Levenshtein Transformer

The Levenshtein Transformer (LevT, Gu, Wang, and Zhao, 2019) is a Transformer model variant that belongs to the aforementioned category of iterative non-autoregressive models. Unlike normal autoregressive sequence models that have only one prediction head to predict the next output words, LevT has two extra prediction heads $A_{del}$ and $A_{ins}$ (along with the word prediction head $A_w$) that predicts deletion and insertion operations based on the output sequence from the previous iteration.

For translation generation, at the $k$-th iteration during decoding, with source-side input $x$ and target-side sequence input from the previous iteration $y^{(k-1)}$ of length $J + 1$, suppose the decoder block output is $\{h_0, h_1, \ldots, h_J\}$. The following predictions and edits will take place in order:

- deletion actions $d_j \in D^{(k)}$:

  \[ p_{del}^{(k)}(d_j \mid x, y^{(k-1)}) = \text{softmax}(A_{del}^T h_j) \]

  for $j \in 0 \ldots J$.

- mask insertion actions $\gamma_j \in S^{(k)}$:

  \[ p_{ins}^{(k)}(\gamma_j \mid x, y') = \text{softmax}(A_{ins}^T [h_{j-1}; h_j]) \]
**CHAPTER 6. IMPROVING QUALITY ESTIMATION MODELS THROUGH LEVENSHTEIN TRAINING**

Figure 6.1: Figure (a) shows an example of TER-style edit labels used as reference for word-level quality estimation task. Figure (b) shows a series of hypothetical Levenshtein Transformer edits that generates the same sequence from the target input. The similarity of these edit operations motivates the study in this paper.

\[
\text{for } j \in 0 \ldots J_1, y' = \mathcal{D}^{(k)}(y^{(k-1)}) \text{ and } J_1 = |y'|.  
\]

- word prediction actions \( w_j \in \mathcal{W}^{(k)}: \)

\[
p_{w}^{(k)}(w_j \mid x, y'') = \text{softmax}(A_w^T h_j) 
\]

\[
\text{for } j \in 0 \ldots J_2, y'' = \mathcal{S}^{(k)}(\mathcal{D}^{(k)}(y^{(k-1)})) \text{ and } J_2 = |y''| - 1.
\]

In the end, \( y^{(k)} = \mathcal{W}^{(k)}(\mathcal{S}^{(k)}(\mathcal{D}^{(k)}(y^{(k-1)}))) \). The iterative process will continue until \( y^{(k)} = y^{(k-1)} \) or a maximum number of iterations is reached.

Figure 6.1 highlights the similarity between the predictions of a LevT decoder and the reference tags of a quality estimation model. It should be specifically noticed that the

\[1\text{Note that } J_1 \text{ is larger than } J_0 \text{ and } J_2 \text{ by one because insertions both before the first word and after the last word are allowed. In the implementation, the hidden states of the boundary tokens marking the start and the end of the sentence will be used to make up for the missing hidden states } h_{-1} \text{ and } h_{J_1}.\]
predictions from the deletion head matches with the word quality labels, while the predictions of the insertion heads matches with the gap quality labels. Hence, even only trained on parallel data without the expensive human post-editing process, a LevT model should have already acquired initial knowledge for post-editing by editing its own intermediate translation output.

After proper adaption of the model to the end task (which we will cover in Section 6.3), our final quality estimation model will use the word deletion head $A_{del}$ to estimate $p(l_j^w | y, x)$ and the mask insertion head $A_{ins}$ to predict $p(l_j^g | y, x)$. Because the end task is to predict binary quality labels for a given translation, our final model only performs one iteration of the above process, and do not use $A_w$ to predict edited words.

It should also be pointed out that using a Levenshtein Transformer translation model as a pre-trained model is similar in spirit to ELECTRA (Clark et al., 2020), which performs pre-training by learning to detect corrupted tokens generated from a masked language model.

### 6.2.3 Pre-trained Model

To achieve optimal performance and take advantage of multilinguality, we would also like to take advantage of large-scale pre-training. Because there is no pre-trained LevT translation model available, we choose to incorporate M2M-100 (Fan et al., 2020b), a
large-scale pre-trained multilingual autoregressive transformer model. Recall that the main architectural difference between LevT and a standard transformer model is the two extra prediction heads on LevT. Hence, to adapt it into a LevT model, we first need to add randomly-initialized extra prediction heads of LevT to the pre-trained checkpoint. These randomly-initialized prediction heads are then trained with the rest of the model on parallel data in order to adapt the autoregressive translation model to a LevT-style non-autoregressive translation model.

6.3 Transfer Learning from Translation to Word-level Quality Estimation

As we pointed out above, by training for the translation task, LevT already acquired some initial knowledge of post-editing. However, there are still some train/inference time mismatches for the word-level QE task regarding (1) the target-side context and (2) the prediction format. In terms of target-side context, during LevT training, the target-side context is a noisy version of the real target sentence in the training set; during inference, the target-side context is a translation output from an NMT system. In terms of the prediction format, during LevT training, we want the model to predict edit tags in the Longest Common Sequence (LCS) format (insertion and deletion) that correct the noisy version of the target
CHAPTER 6. IMPROVING QUALITY ESTIMATION MODELS THROUGH LEVENSHTEIN TRAINING

sentence; during inference, we want the model to predict Levenshtein format edit tags (insertion, deletion and substitution) that correspond to human post-editing of NMT outputs.

To address such mismatch, we adopt a two-step transfer learning scheme. Like the estimator finetuning for PredEst framework, for both stages of transfer learning, we need translation triplets (source, MT output, post-edited output) to perform finetuning. However, in practice, human post-editing resources are quite scarce. Hence, like some previous work (Lee, 2020b; Wang et al., 2020a), we start by performing transfer learning on synthetic translation triplets, followed by real translation triplets constructed with human post-editing.

6.3.1 Synthetic Data Construction

We explore five different methods for translation triplet synthesis:

- **src-mt-ref** Take a parallel dataset \((src, ref)\) and translate the source sentence with an MT model \((mt)\).

- **bt-rt-tgt** Take a target-side monolingual dataset \((tgt)\). Translate the target sentence with a backward MT model \((bt)\) and then translate \(bt\) again with an forward MT model, thus creating a round-trip translated output \((rt)\). Use \(rt\) as the MT output and the original \(tgt\) as the pseudo post-edited output.

- **src-mt1-mt2** Take a source-side monolingual dataset \((src)\). Translate the source
CHAPTER 6. IMPROVING QUALITY ESTIMATION MODELS THROUGH LEVENShteIN TRAINING

sentence with a weaker MT model (mt1) and a stronger MT model (mt2). Use mt1 as the MT output and mt2 as the pseudo post-edited output.

• src-rt-ft: Take a parallel corpus and translate its source side and use it as the pseudo post-edited output (ft), and round-trip translate its target side (rt) as the MT output in the translation triplet.

• mvppe Take the source-side of a parallel dataset (src). Translate the source sentence with a multilingual MT model as the MT output (mt) and build a pseudo post-edited output (pe) with multiview pseudo post-edit (MVPPE) decoding, as described below.

Multiview Pseudo Post-Editing (MVPPE) Inspired by Thompson and Post (2020b) which used a multilingual translation system as a zero-shot paraphraser, we propose a novel pseudo post-editing method to build synthetic post-editing dataset from parallel corpus (src, tgt). The first step is to translate the source side of the parallel corpus with a multilingual translation system as the MT output (mt) in the triplet. We then generate the pseudo-post-edited output by ensembling two different views of the same model. These two views are:

• the translation output distribution \( p_t(\text{pe} \mid \text{src}) \), with src as the model input;

• the paraphrase output distribution \( p_p(\text{pe} \mid \text{tgt}) \), with tgt as the model input.
Note that both views will create a distribution in the target language space, which can be ensembled in the same way as standard MT model ensembles, forming an interpolated distribution:

\[ p(\text{pe} | \text{src}, \text{tgt}) \equiv \lambda_t p_t(\text{pe} | \text{src}) + \lambda_p p_p(\text{pe} | \text{tgt}) \]  

(6.1)

with \( \lambda_t \) and \( \lambda_p \) as the interpolation weights. Similarly, beam search can also be performed on top of the ensemble. The intuition behind this idea is that such ensemble should create a target sentence that is semantically equivalent to the target side of the parallel corpus, while being close to the original MT output as much as possible, imitating the way humans perform the post-editing task.

### 6.3.2 Compatibility with Subwords

To the best of our knowledge, previous work on word-level quality estimation either builds models that directly output word-level tags (Lee, 2020b; Hu et al., 2020; Moura et al., 2020) or uses simple heuristics to re-assign word-level tags to the first subword token (Wang et al., 2020a). Since LevT predicts edits on a subword-level starting from translation training, we need to: (1) for inference, convert subword-level tags predicted by the model to word-level tags for evaluation, and (2) for both finetuning stages, build
CHAPTER 6. IMPROVING QUALITY ESTIMATION MODELS THROUGH LEVENSHTEIN TRAINING

subword-level reference tags.

Inference Algorithm 1 shows our procedure to convert subword-level tags produced by the model to word-level tags. The simple heuristic used in this algorithm is the following: if one or more of the subwords corresponding to a word is labeled as BAD, then the whole word is labeled as BAD, and vice versa.

Algorithm 1: Conversion of subword-level tags to word-level tags

| **Input:** | subword-level token sequence $y^{sw}$, word-level token sequence $y^{w}$, subword-level tag sequence $q^{sw}$ |
| **Output:** | word-level tag sequence $q^{w}$ |
| $q^{w} \leftarrow []$; |
| **for each word** $w_k$ in $y^{w}$ **do** |
| find subword index span $(s_k, e_k)$ in $y^{sw}$ that corresponds to $w_k$; |
| $q^{sw}_{k} \leftarrow$ subword-level translation and gap tags within span $(s_k, e_k)$; |
| $g^{sw}_{s_k} \leftarrow$ subword-level gap tag before span $(s_k, e_k)$; \hspace{1cm} // $g^{sw}_{s_k-1} \in q^{sw}$ |
| if $\forall q^{sw}_{k}$ are OK then |
| $q^{w} += [g^{sw}_{s_k}, \text{OK}]$; |
| else |
| $q^{w} += [g^{sw}_{s_k}, \text{BAD}]$; |
| end |
| $q^{w} += [q^{sw}[-1]]$; \hspace{1cm} // add ending gap tag |
| return $q^{w}$; |

Finetuning Intuitively, we can build subword-level tag reference by directly using TER alignments built on MT and post-edited text after subword tokenization (BPE, sentencepiece, etc.), which we refer to as naive subword-level reference from now on. However, since subword tokenization interacts with the word order error detection (sometimes referred to as “shifts”) in the TER computation, the naive subword-level references computed
Algorithm 2: Construction of heuristic subword-level tags

**Input:** subword-level token sequence $y^{sw}$, word-level token sequence $y^{w}$, naive subword-level tag sequence $q^{sw}$, word-level tag sequence $q^{w}$

**Output:** heuristic subword-level tag sequence $q^{sw}\hat{}$

$q^{sw}\hat{} \leftarrow []$;

for each word $w_k$ in $y^{w}$ do

- find subword index span $(s_k, e_k)$ in $y^{sw}$ that corresponds to $w_k$;
- $q^{sw}_k \leftarrow$ subword-level translation and gap tags within span $(s_k, e_k)$;
- $t_k^w \leftarrow$ word-level translation tag for $w_k$; // $t_k^w \in q^{w}$
- $g_k^w \leftarrow$ word-level gap tag before $w_k$; // $g_k^w \in q^{w}$
- $q^{sw} += [g_k^w]$; // copy left gap tag
- $n = |q^{sw}_k|$; // # subwords for $w_k$

if $t_k^w$ is OK then

- /* word is OK, so all subwords and gaps between them should be OK */
- $q^{sw}\hat{} += [OK] \ast (2n - 1)$

else if $\exists$ BAD in $q^{sw}_k$ then

- /* no conflict between subword-level tag and word-level tag - copy $q^{sw}_k$ as-is */
- $q^{sw}\hat{} += q^{sw}_k$;

else

- /* subword-level tag disagrees with word-level tag - force it as all-BAD to guarantee perfect conversion */
- $q^{sw}\hat{} += [BAD] \ast (2n - 1)$

end

$q^{sw}\hat{} += [q^{w}[-1]]$; // add ending gap tag

return $q^{sw}\hat{}$;
CHAPTER 6. IMPROVING QUALITY ESTIMATION MODELS THROUGH LEVENSHTEIN TRAINING

this way cannot be perfectly converted back to the word-level references using Algorithm 1 introduced above.\(^2\) Since our model will eventually be evaluated on word-level tags references, finetuning against such naive references is not ideal.

We propose to build an alternative reference that allows perfect conversion back to their word-level counterparts. The high-level idea is to heuristically interpolate the word-level references and the naive subword-level references to ensure the interpolated subword-level tag reference can be perfectly converted back to the word-level references. We refer to this interpolated version as \textit{heuristic subword-level reference}. The detailed algorithm is shown in Algorithm 2.

6.3.3 Label Imbalance

Like several previous work (Lee, 2020a; Wang et al., 2020b; Moura et al., 2020), we also observed that the translation errors are often quite scarce, thus creating a skewed label distribution over the OK and BAD labels. Since it is critical for the model to reliably predict both classes of labels, we introduce an extra hyperparameter \(\mu\) in the loss function that allows us to upweight the words that are classified with BAD tags in the reference.

\[
\mathcal{L} = \mathcal{L}_{OK} + \mu \mathcal{L}_{BAD}
\]

\(^2\)A preliminary analysis shows that the difference is approximately 10%.
CHAPTER 6. IMPROVING QUALITY ESTIMATION MODELS THROUGH LEVENSHTEIN TRAINING

6.3.4 Ensemble

For each binary label prediction made by the model, the model will give a score \( p(l^w_j \mid y, x) \) or \( p(l^g_j \mid y, x) \) depending on whether the prediction is for a translation word or gap. These scores are binarized into discrete labels during post-processing. To ensemble predictions from \( k \) models, we perform a linear combination of the scores for each label. Take the word labels for example:

\[
P_E(l^w_j \mid y, x) = \lambda_1 p_1(l^w_j \mid y, x) + \lambda_2 p_2(l^w_j \mid y, x) + \cdots + \lambda_k p_k(l^w_j \mid y, x) \tag{6.2}
\]

and similarly for gaps. To determine the optimal interpolation weights, we optimize on the development set with the task evaluation metric as objective. Because most of the evaluation metrics for the quality estimation task are not implemented in a way such that gradient information can be easily obtained, we experimented with two gradient-free optimization method: the Powell method (Powell, 1964) and the Nelder-Mead method (Nelder and Mead, 1965), both as implemented in SciPy (Virtanen et al., 2020). We found that the Nelder-Mead method finds better optimum on the development set while also leading to better performance on the devtest dataset (not involved in optimization). Hence, we use the Nelder-Mead optimizer for all of our final submissions with ensembles. We set the initial points of Nelder-Mead optimization to be the vertices of the standard simplex in the
6.4 Experiments on WMT 2020 Dataset

The first set of experiments is based on the setup of WMT 2020 quality estimation shared task (Specia et al., 2020a)\textsuperscript{3}. The results are reported under two settings: the constrained setting and the unconstrained setting. In the constrained setting, we focus on the data efficiency of our model and use only the human-labeled dataset, the NMT model, and the parallel data (used to train the NMT model and the PredEst baseline) provided by the shared task, with neither large-scale pre-trained model nor synthetic data finetuning. As the PredEst model comparison, we will be comparing with the OpenKiwi baseline system from WMT 2020 quality estimation shared task. Note that unlike the OpenKiwi systems we built in the previous chapter or for our WMT 2021 experiments in the next section, these OpenKiwi systems train their own predictor on the constrained parallel datasets instead of using a pre-trained encoder. On the other hand, in the unconstrained setting, we additionally use some extra resources we have access to. Correspondingly, we compare our results with the winning systems from the shared task. Table 6.1 shows some statistics of the data we use in our experiments.

\textsuperscript{3}Note that this is a slightly out-dated version compared to the WMT 2021 setup used for the experiments in Chapter 5. We will also report results of Levenshtein-based quality estimation based on WMT 2021 setup in the next section.
CHAPTER 6. IMPROVING QUALITY ESTIMATION MODELS THROUGH LEVENSHTEIN TRAINING

6.4.1 Data Setup

**LevT Training**  Under the constrained setting, only the data provided by the shared task is used. For the unconstrained setting, we use WMT20 en-de parallel data to train a LevT model for en-de. For en-zh, we use the same dataset because it is already close to the WMT data scale. We also experiment with the M2M-100-small initialization (Fan et al., 2020b, 418M parameters) as described in Section 6.2.3. Note that M2M-100 directly applies sentencepiece on untokenized data, a tokenization scheme that is incompatible with the shared task setting. For our experiments with M2M-100-small, we proceed with applying sentencepiece on tokenized data during finetuning. We also experiment with synthetic data finetuning with different data synthesis methods on en-de language pair, while for en-zh, because we don’t have access to an extra high-quality MT system, we only experiment with mvPPE method.

We preprocess our data by first tokenizing with Moses tokenizer, and then applying subword segmentation. For all the LevT models without M2M-100 initialization, we use the same BPE model and source/target-side vocabulary as the official NMT checkpoint provided by the WMT20 quality estimation shared task. For models with M2M-100 initialization, we use the M2M-100 sentencepiece model.

**Synthetic Finetuning**  For all en-de experiments, we took the first 1 million lines from the
CHAPTER 6. IMPROVING QUALITY ESTIMATION MODELS THROUGH LEVENSHTEIN TRAINING

<table>
<thead>
<tr>
<th># sentence pairs/triplets</th>
</tr>
</thead>
<tbody>
<tr>
<td>shared task en-de parallel</td>
</tr>
<tr>
<td>shared task en-zh parallel</td>
</tr>
<tr>
<td>WMT20 en-de parallel</td>
</tr>
<tr>
<td>en-de src-mt-ref synthetic</td>
</tr>
<tr>
<td>en-de src-mt1-mt2 synthetic</td>
</tr>
<tr>
<td>en-de bt-rt-tgt synthetic</td>
</tr>
<tr>
<td>en-de mvppe synthetic</td>
</tr>
<tr>
<td>en-zh mvppe synthetic</td>
</tr>
<tr>
<td>shared task en-de human PE train</td>
</tr>
<tr>
<td>shared task en-zh human PE train</td>
</tr>
<tr>
<td>shared task en-de human PE dev</td>
</tr>
<tr>
<td>shared task en-zh human PE dev</td>
</tr>
<tr>
<td>shared task en-de human PE test</td>
</tr>
<tr>
<td>shared task en-zh human PE test</td>
</tr>
</tbody>
</table>

Table 6.1: Data Statistics for Our WMT 2020 Experiments

Europarl corpus and conduct data synthesis. For this group of experiments, we experimented with four data synthesis methods:

- **src-mt-ref** We translate the source side of the parallel data with the NMT system provided by the shared task organizer.

- **bt-rt-tgt** Both the back-translation and the round-trip translation are performed with M2M-100-mid (1.2B) model.

- **src-mt1-mt2** We take the source side of the parallel data and translate it with the NMT system from the shared task (weaker system, mt1) and the Facebook winning system for the WMT19 en-de news translation (Ng et al., 2019, stronger system, mt2).
We remove all the cases where \( mt1 \) and \( mt2 \) are identical.

- **mvppe** The MVPPE decoding is conducted with M2M-100-mid (1.2B) model.

For en-zh experiments, we take the shared task en-zh parallel data but exclude the UN data for MVPPE data synthesis. The same multilingual translation model is used. We also experimented with using larger synthetic data for en-de with some synthesis method, but didn’t observe a significant performance difference compared to this smaller dataset.

**Human Annotation Finetuning** We follow the data split for human post-edited data as determined by the WMT 2020 shared task organizers. Note that there are two different versions of human post-editing for the same WMT 2020 data (including the training, development and test set). Most of our experiments were run before the most up-to-date version was released, so all of our human annotation finetuning are based on the first version\(^4\).

**Reference Tag Generation** We implemented another TER computation tool\(^5\) to generate the word-level and subword-level tags that we use as the reference for finetuning, but stick to the original reference tags in the test set for evaluation to avoid potential result mismatch.

---

\(^4\) Accessible through https://github.com/sheffieldnlp/mlqe-pe commit 0c0773a.

\(^5\) https://github.com/marian-nmt/moses-scorers
6.4.2 Model Setup

Under the constrained setting, we use the NMT checkpoint supplied by the shared task to generate the knowledge distillation data for LevT translation training, both for en-de and en-zh. Under the unconstrained setting, for en-de, we use the Facebook winning system for the WMT19 en-de news translation, and for en-zh, we use our own transformer-base en-zh model trained on WMT17 en-zh data.

All of our implementations are based on the Fairseq toolkit. We use the same hyperparameter for LevT translation model training as the document provided in Fairseq\(^6\). For both synthetic and human post-edited data finetuning, we use Adam optimizer with a learning rate 2e-5 with warmup (4000 updates for synthetic finetuning and 2000 for human post-edited data finetuning), and we use the shared task development set to select the best checkpoint.

For all the mvppe experiments, we use \(\lambda_t = 2.0\) and \(\lambda_p = 1.0\), after doing a grid search over \(\lambda_t = \{1.0, 2.0, 3.0\}\) and \(\lambda_p = \{1.0, 1.2, 1.5\}\) with a goal to match the TER distribution of human post-editing obtained from the en-de human PE dev data.

For this set of experiments, we do not explore ensemble and always report results from single models. We also keep label imbalance factor \(\mu = 1\) for all reported results.

---

\(^6\)https://github.com/pytorch/fairseq/blob/master/examples/nonautoregressive_translation/README.md
CHAPTER 6. IMPROVING QUALITY ESTIMATION MODELS THROUGH LEVENSHTEIN TRAINING

<table>
<thead>
<tr>
<th></th>
<th>MCC</th>
<th>F1-OK</th>
<th>F1-BAD</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>en-de</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OpenKiwi</td>
<td>0.358</td>
<td>0.879</td>
<td>0.468</td>
</tr>
<tr>
<td>LevT w/o KD</td>
<td>0.441</td>
<td>0.926</td>
<td>0.498</td>
</tr>
<tr>
<td>LevT</td>
<td><strong>0.477</strong></td>
<td><strong>0.929</strong></td>
<td><strong>0.535</strong></td>
</tr>
<tr>
<td><strong>en-zh</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OpenKiwi</td>
<td>0.509</td>
<td>0.849</td>
<td>0.658</td>
</tr>
<tr>
<td>LevT</td>
<td><strong>0.629</strong></td>
<td><strong>0.885</strong></td>
<td><strong>0.741</strong></td>
</tr>
</tbody>
</table>

Table 6.2: Constrained setting. All LevT models here are transformer-base models. F1-OK and F1-BAD are F1 scores of the OK and BAD tags, respectively. Higher is better for all metrics in this table.

### 6.4.3 Evaluation Setup

Our evaluations are done with the official evaluation scripts\(^7\) from the shared task. The script computes the Matthews Correlation Coefficient (MCC, Matthews, 1975) as well as F1 score of the OK and BAD tags (F1-OK and F1-BAD in the result tables). The results we report in this section are on the WMT 2020 test set with the first version of human post-editing, which we refer to as test20-v1.

### 6.4.4 Results

Table 6.2 shows results under the data-constrained setting. Even without knowledge distillation (KD) during LevT training, our model already scores much higher than the baseline OpenKiwi system. When training with KD data generated with the shared task

\(^7\)[https://github.com/sheffieldnlp/qe-eval-scripts](https://github.com/sheffieldnlp/qe-eval-scripts)
CHAPTER 6. IMPROVING QUALITY ESTIMATION MODELS THROUGH LEVENSHTEIN TRAINING

<table>
<thead>
<tr>
<th>Init.</th>
<th>LevT</th>
<th>Data Synth.</th>
<th>MCC</th>
<th>F1-OK</th>
<th>F1-BAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>en-de</td>
<td>N</td>
<td>base N</td>
<td>0.539</td>
<td>0.925</td>
<td>0.613</td>
</tr>
<tr>
<td>N</td>
<td>base src-mt-ref</td>
<td>0.542</td>
<td>0.925</td>
<td>0.616</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>base bt-rt-tgt</td>
<td>0.535</td>
<td>0.925</td>
<td>0.609</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>base mvppe</td>
<td>0.548</td>
<td>0.926</td>
<td>0.620</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>base src-mt1-mt2</td>
<td>0.549</td>
<td>0.926</td>
<td>0.622</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>big N</td>
<td>0.551</td>
<td>0.927</td>
<td>0.623</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>big src-mt1-mt2</td>
<td>0.562</td>
<td>0.939</td>
<td>0.617</td>
<td></td>
</tr>
<tr>
<td>M2M</td>
<td>418M N</td>
<td>0.583</td>
<td>0.932</td>
<td>0.650</td>
<td></td>
</tr>
<tr>
<td>M2M</td>
<td>418M src-mt1-mt2</td>
<td>0.589</td>
<td>0.934</td>
<td>0.654</td>
<td></td>
</tr>
<tr>
<td>en-zh</td>
<td>N</td>
<td>base N</td>
<td>0.629</td>
<td>0.885</td>
<td>0.741</td>
</tr>
<tr>
<td>N</td>
<td>big N</td>
<td>0.625</td>
<td>0.885</td>
<td>0.738</td>
<td></td>
</tr>
<tr>
<td>M2M</td>
<td>418M N</td>
<td>0.633</td>
<td>0.884</td>
<td>0.744</td>
<td></td>
</tr>
<tr>
<td>M2M</td>
<td>418M mvppe</td>
<td>0.646</td>
<td>0.892</td>
<td>0.752</td>
<td></td>
</tr>
<tr>
<td>WMT20</td>
<td>en-de best</td>
<td>0.597</td>
<td>0.935</td>
<td>0.662</td>
<td></td>
</tr>
<tr>
<td>en-zh best</td>
<td>0.610</td>
<td>0.887</td>
<td>0.723</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6.3: Unconstrained setting. base and big stand for the transformer-base and transformer-big architecture. 418M is the M2M-100-small model.

NMT system (trained on the same parallel data), the advantage of the LevT expands even more. This shows that our proposal to build a word-level quality estimation system from a LevT translation model has higher data efficiency than the widely adopted PredEst approach used by the OpenKiwi baseline.

Table 6.3 shows results under the unconstrained setting. We first notice that fine-tuning with synthetic data before human post-edited data almost always helps. With the transformer-base model on en-de language pair, we experimented with all four data synthesis methods,
and we find that src-mt1-mt2 performs the best, closely followed by mvppe. This might be related to the fact that all LevT models are trained with KD data. Because of this, the model is better posed to fit synthesized data with MT-like output as pseudo post-edited data, instead of human-generated translations. Also, initializing with the M2M-100 model is helpful despite the tokenization scheme mismatch, although the performance gain is much more modest on en-zh language pair. This is possibly influenced by the relatively low translation quality of M2M-100 model on en-zh language pair, as pointed out by Fan et al. (2020b).

With all our techniques applied, our best Target MCC result is only slightly behind the winning system on en-de language pair, while being significantly better than the winning system on en-zh language pair. Most notably, for the en-zh language pair, even our smallest LevT model is able to beat the state of the art. It should also be pointed out that all of our results are achieved without any model ensemble, and our pre-trained model architecture is just a transformer-big counterpart, while other participating teams deployed models finetuned from the larger XLM-RoBERTa-large encoder.

To confirm that each component of our training scheme is necessary, we conducted a comprehensive ablation study on the en-de language pair, shown in Table 6.4. The upper part of the table demonstrates that LevT training is necessary, and we do so by conducting the finetuning directly on M2M-100-small initialization. Despite the strength of the M2M-100 model as a translation model, there is still a significant performance drop
CHAPTER 6. IMPROVING QUALITY ESTIMATION MODELS THROUGH LEVENSHTEIN TRAINING

<table>
<thead>
<tr>
<th>Ablation Configuration</th>
<th>MCC</th>
<th>F1-OK</th>
<th>F1-BAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>best</td>
<td>0.589</td>
<td>0.934</td>
<td>0.654</td>
</tr>
<tr>
<td>-LevT</td>
<td>0.555</td>
<td>0.938</td>
<td>0.610</td>
</tr>
<tr>
<td>-LevT +lang-adapt</td>
<td>0.565</td>
<td>0.930</td>
<td>0.635</td>
</tr>
<tr>
<td>-LevT -synth.</td>
<td>0.321</td>
<td>0.915</td>
<td>0.380</td>
</tr>
<tr>
<td>-LevT -synth. +lang-adapt</td>
<td>0.451</td>
<td>0.946</td>
<td>0.498</td>
</tr>
<tr>
<td>-m2m</td>
<td>0.562</td>
<td>0.939</td>
<td>0.617</td>
</tr>
<tr>
<td>-m2m -KD</td>
<td>0.526</td>
<td>0.933</td>
<td>0.589</td>
</tr>
<tr>
<td>-m2m -heuristic tag</td>
<td>0.551</td>
<td>0.936</td>
<td>0.610</td>
</tr>
<tr>
<td>-m2m -synth.</td>
<td>0.551</td>
<td>0.927</td>
<td>0.623</td>
</tr>
<tr>
<td>-m2m -synth. -heuristic tag</td>
<td>0.539</td>
<td>0.925</td>
<td>0.613</td>
</tr>
</tbody>
</table>

Table 6.4: Ablation analysis. All results trained with synthetic data in this table use the src-\texttt{mt1-mt2} data synthesis method. +lang-adapt stands for adding an extra autoregressive MT training step using the same parallel training data as LevT training, so the M2M-100 model is adapted to translating a specific language pair.

without LevT Training, and more so without synthetic finetuning. To rule out the effect of bilingual knowledge introduced with LevT training, we also experimented with adapting the M2M-100 model (+lang-adapt in Table 6.4) with the same parallel data used for LevT training, but the performance gap remains. On the other hand, the lower part of the table highlights the effect of various other training techniques, where we use the best system without M2M-100-small initialization as the base. We can conclude that KD is crucial for optimal performance and that finetuning with heuristic subword-level tag reference is responsible for a small but consistent performance improvement.
CHAPTER 6. IMPROVING QUALITY ESTIMATION MODELS THROUGH
LEVENSHTEIN TRAINING

Table 6.5: Data Source and Statistics of Parallel Datasets Used in Our WMT 2021 Experiments

<table>
<thead>
<tr>
<th>Language Pair</th>
<th>Data Source</th>
<th>Sentence Pairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>English-German</td>
<td>WMT20 en-de parallel data</td>
<td>44.2M</td>
</tr>
<tr>
<td>English-Chinese</td>
<td>shared task en-zh parallel</td>
<td>20.3M</td>
</tr>
<tr>
<td>Romanian-English</td>
<td>shared task ro-en parallel</td>
<td>3.09M</td>
</tr>
<tr>
<td>Russian-English</td>
<td>shared task ru-en parallel</td>
<td>2.32M</td>
</tr>
<tr>
<td>Estonian-English</td>
<td>shared task et-en parallel</td>
<td>880K</td>
</tr>
<tr>
<td>Estonian-English</td>
<td>shared task et-en parallel + NewsCrawl 14-17</td>
<td>3.42M</td>
</tr>
<tr>
<td>Nepalese-English</td>
<td>shared task ne-en parallel</td>
<td>498K</td>
</tr>
</tbody>
</table>

6.5 Experiments on WMT 2021 Dataset

In this section, we describe configurations of our system submission for the WMT 2021 quality estimation shared task and report results on the devtest dataset we used for our system development. While the evaluation setup is mostly the same, the 2021 version of the shared task provided updated human annotations and expanded to more language pairs. The experiment results we report here cover 6 of them. Apart from further confirming the observations we made on WMT 2020 datasets, we also explore configurations such as label imbalance factors and ensemble.

6.5.1 Data Setup

**LevT Training** We used the same parallel data that was used to train the MT system in the shared task, except for the en-de, et-en language pairs. For en-de language pair, we use the
CHAPTER 6. IMPROVING QUALITY ESTIMATION MODELS THROUGH LEVENSHTEIN TRAINING

larger parallel data from the WMT20 news translation shared task. For et-en language pair, we experiment with augmenting with the News Crawl Estonian monolingual data from 2014 to 2017, which was inspired by Zhou and Keung (2020). Since we are now experimenting with more language pairs, we also explored training a multilingual LevT model for all the to-English language pairs. To train this model, we simply concatenate the data from ro-en, ru-en, et-en (without the monolingual augmentation) and ne-en. The resulting data sizes are summarized in Table 6.5.

Following the setup in Gu, Wang, and Zhao (2019), we conduct sequence-level knowledge distillation during training for all language pairs except for ne-en. For en-de, the knowledge distillation data is generated by the WMT19 winning submission for that language pair from Facebook (Ng et al., 2019). For en-zh, we train our own en-zh autoregressive model on the parallel data from the WMT17 news translation shared task. For the other language pairs, we use the decoding output from M2M-100-mid (1.2B parameters) model to perform knowledge distillation.

Synthetic Finetuning We always conduct data synthesis based on the same parallel data that was used to train the LevT translation model. For the only language pair (en-de) where we applied the src-mt1-mt2 synthetic finetuning for the shared task submission, we again use the WMT19 Facebook’s winning system (Ng et al., 2019) to generate the higher-quality

---
8The exception was motivated by the poor quality of translations we obtained from the M2M-100 model.
CHAPTER 6. IMPROVING QUALITY ESTIMATION MODELS THROUGH
LEVENSHTEIN TRAINING

translation mt2, and the system provided by the shared task to generate the MT output in the
pseudo translation triplet mt1. For all other combinations of translation directions, language
pairs and MVPPE decoding, we use the M2M-100-mid (1.2B parameters) model.

Human Annotation Finetuning We follow the data split for human post-edited data as
determined by the WMT 2021 shared task organizers, but use the test data from WMT 2020
as the devtest dataset. Note that there are two different versions of human annotation for the
same WMT 2020 test data. For results in this section, we use the updated version denoted
as test20-v29.

6.5.2 Model Setup

Most of the model configurations are the same as what we used for WMT 2020 exper-
iments except for the label imbalance factor. For human annotation finetuning, we also
experiment with label balancing factor $\mu = 1.0$ and $\mu = 3.0$ for each language pair and pick
the one that works the best on devtest data, while for synthetic finetuning we keep $\mu = 1.0$
because early experiments indicate that using $\mu = 3.0$ at this stage is not helpful.

For pre-submission developments, same as in Chapter 5, we built OpenKiwi-XLM
baselines (Kepler et al., 2019b) following their xlmroberta.yaml recipe. Keep in mind
that because this baseline model is initialized with a smaller XLM-Roberta-base model

9Accessible through https://github.com/sheffieldnlp/mlqe-pe commit db898c0.
CHAPTER 6. IMPROVING QUALITY ESTIMATION MODELS THROUGH
LEVENSHTEIN TRAINING

(281M parameters) compared to our M2M-100-small initialization (484M parameters), the
performance comparison is not a strict one.

We also find that it is critical to build ensembles from models that yield diverse yet
high-quality outputs. Specifically, we notice that ensembles from multiple checkpoints of a
single experimental run are not helpful. Hence, for each language pair, we select 2-8 models
with different training configurations that also have the highest performance to build our
final ensemble model.

6.5.3 Devtest Results

Our system development results on test20-v2 devtest data are shown in Table 6.6\textsuperscript{10}.
For all language pairs, our systems can outperform the OpenKiwi baseline based upon the
pre-trained XLM-RoBERTa-base encoder. Among these language pairs, the benefit of LevT
is most significant on the language pairs with a large amount of available parallel data. Such
behavior is expected, because the less parallel data we have, the less knowledge we can
extract from the LevT training process. Furthermore, the lack of good quality knowledge
distillation data in the low-resource language pairs also increases this performance gap.

In terms of comparison between multilingual and bilingual models for to-English lan-

\textsuperscript{10}Note that the results on en-zh also reflect a crucial bug fix on our TER computation tool that we added
after the system submission deadline. Hence the results shown here are from a different system as in the
official shared task results. The bug fix should not affect the results of the other language pairs.
<table>
<thead>
<tr>
<th>Configuration</th>
<th>Stage 2</th>
<th>Stage 3</th>
<th>Target MCC</th>
</tr>
</thead>
<tbody>
<tr>
<td>en-de OpenKiwi</td>
<td>N</td>
<td>default</td>
<td>0.337</td>
</tr>
<tr>
<td>en-de bilingual best</td>
<td>src-mlt1-mlt2</td>
<td>(\mu = 1.0)</td>
<td>0.500</td>
</tr>
<tr>
<td>en-de ensemble</td>
<td>N/A</td>
<td>N/A</td>
<td>0.504</td>
</tr>
<tr>
<td>en-zh OpenKiwi</td>
<td>N</td>
<td>default</td>
<td>0.421</td>
</tr>
<tr>
<td>en-zh bilingual best</td>
<td>mvppe</td>
<td>(\mu = 1.0)</td>
<td>0.459</td>
</tr>
<tr>
<td>en-zh ensemble</td>
<td>N/A</td>
<td>N/A</td>
<td>0.466</td>
</tr>
<tr>
<td>ro-en OpenKiwi</td>
<td>N</td>
<td>default</td>
<td>0.556</td>
</tr>
<tr>
<td>ro-en bilingual best</td>
<td>src-rt-ft</td>
<td>(\mu = 1.0)</td>
<td>0.604</td>
</tr>
<tr>
<td>ro-en multilingual best</td>
<td>N</td>
<td>(\mu = 1.0)</td>
<td>0.612</td>
</tr>
<tr>
<td>ro-en ensemble</td>
<td>N/A</td>
<td>N/A</td>
<td>0.633</td>
</tr>
<tr>
<td>ru-en OpenKiwi</td>
<td>N</td>
<td>default</td>
<td>0.279</td>
</tr>
<tr>
<td>ru-en bilingual best</td>
<td>src-rt-ft</td>
<td>(\mu = 3.0)</td>
<td>0.316</td>
</tr>
<tr>
<td>ru-en multilingual best</td>
<td>N</td>
<td>(\mu = 3.0)</td>
<td>0.339</td>
</tr>
<tr>
<td>ru-en ensemble</td>
<td>N/A</td>
<td>N/A</td>
<td>0.349</td>
</tr>
<tr>
<td>et-en OpenKiwi</td>
<td>N</td>
<td>default</td>
<td>0.503</td>
</tr>
<tr>
<td>et-en bilingual best</td>
<td>N</td>
<td>(\mu = 3.0)</td>
<td>0.556</td>
</tr>
<tr>
<td>et-en bilingual best (w/ aug)</td>
<td>N</td>
<td>(\mu = 3.0)</td>
<td>0.548</td>
</tr>
<tr>
<td>et-en multilingual best</td>
<td>N</td>
<td>(\mu = 3.0)</td>
<td>0.533</td>
</tr>
<tr>
<td>et-en ensemble</td>
<td>N/A</td>
<td>N/A</td>
<td>0.575</td>
</tr>
<tr>
<td>ne-en OpenKiwi</td>
<td>N</td>
<td>default</td>
<td>0.664</td>
</tr>
<tr>
<td>ne-en bilingual best</td>
<td>N</td>
<td>(\mu = 3.0)</td>
<td>0.677</td>
</tr>
<tr>
<td>ne-en multilingual best</td>
<td>N</td>
<td>(\mu = 3.0)</td>
<td>0.681</td>
</tr>
<tr>
<td>ne-en ensemble</td>
<td>N/A</td>
<td>N/A</td>
<td>0.688</td>
</tr>
</tbody>
</table>

Table 6.6: Target MCC results on test20-v2 dataset for all language pairs we submitted systems for. Stage 2 stands for synthetic finetuning (where N stands for not performing this stage). Stage 3 stands for human annotation finetuning. \(\mu\) stands for the label balancing factor.
CHAPTER 6. IMPROVING QUALITY ESTIMATION MODELS THROUGH LEVENSHTEIN TRAINING

guage pairs, the results are mixed, with the multilingual model performing significantly better for ru-en language pair, but significantly worse for et-en language pair. Finally, our Nelder-Mead ensemble further improves the result by a small but steady margin.

<table>
<thead>
<tr>
<th></th>
<th>Target MCC</th>
<th>F1-OK</th>
<th>F1-BAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>0.489</td>
<td>0.955</td>
<td>0.533</td>
</tr>
<tr>
<td>src-mt-ref</td>
<td>0.493</td>
<td>0.955</td>
<td>0.537</td>
</tr>
<tr>
<td>src-mt1-mt2</td>
<td><strong>0.500</strong></td>
<td>0.956</td>
<td><strong>0.544</strong></td>
</tr>
<tr>
<td>bt-rt-tgt</td>
<td>0.490</td>
<td>0.956</td>
<td>0.534</td>
</tr>
<tr>
<td>src-rt-ft</td>
<td>0.494</td>
<td>0.956</td>
<td>0.538</td>
</tr>
<tr>
<td>mvpe</td>
<td><strong>0.500</strong></td>
<td><strong>0.960</strong></td>
<td>0.540</td>
</tr>
</tbody>
</table>

Table 6.7: Analysis of different data synthesis methods on en-de language pair. All models here are initialized with M2M-100-small.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Stage 2</th>
<th>Stage 3</th>
<th>Target MCC</th>
<th>F1-OK</th>
<th>F1-BAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>ro-en multilingual</td>
<td>N</td>
<td>$\mu = 1.0$</td>
<td><strong>0.612</strong></td>
<td>0.949</td>
<td><strong>0.659</strong></td>
</tr>
<tr>
<td>ro-en multilingual</td>
<td>mvpe</td>
<td>$\mu = 1.0$</td>
<td>0.611</td>
<td><strong>0.951</strong></td>
<td><strong>0.659</strong></td>
</tr>
<tr>
<td>ro-en multilingual</td>
<td>src-mt1-mt2 (Bing mt2)</td>
<td>$\mu = 1.0$</td>
<td>0.585</td>
<td>0.936</td>
<td>0.630</td>
</tr>
<tr>
<td>ro-en bilingual (Bing KD)</td>
<td>N</td>
<td>$\mu = 1.0$</td>
<td>0.581</td>
<td>0.949</td>
<td>0.632</td>
</tr>
<tr>
<td>ro-en bilingual (Bing KD)</td>
<td>src-mt1-mt2 (Bing mt2)</td>
<td>$\mu = 1.0$</td>
<td>0.568</td>
<td>0.938</td>
<td>0.619</td>
</tr>
<tr>
<td>et-en bilingual</td>
<td>N</td>
<td>$\mu = 3.0$</td>
<td>0.548</td>
<td>0.914</td>
<td>0.622</td>
</tr>
<tr>
<td>et-en bilingual</td>
<td>mvpe</td>
<td>$\mu = 3.0$</td>
<td>0.544</td>
<td><strong>0.929</strong></td>
<td>0.615</td>
</tr>
<tr>
<td>et-en bilingual</td>
<td>src-mt1-mt2 (Bing mt2)</td>
<td>$\mu = 3.0$</td>
<td><strong>0.563</strong></td>
<td>0.919</td>
<td><strong>0.634</strong></td>
</tr>
<tr>
<td>et-en bilingual (Bing KD)</td>
<td>N</td>
<td>$\mu = 3.0$</td>
<td>0.557</td>
<td>0.918</td>
<td>0.629</td>
</tr>
<tr>
<td>et-en bilingual (Bing KD)</td>
<td>src-mt1-mt2 (Bing mt2)</td>
<td>$\mu = 3.0$</td>
<td>0.559</td>
<td>0.916</td>
<td>0.631</td>
</tr>
</tbody>
</table>

Table 6.8: Analysis of src-mt1-mt2 and mvpe method on ro-en and et-en language pair.
6.5.4 Analysis

In the previous section, we have already conducted comprehensive ablation studies for the LevT training step, the heuristic subword-level reference tag, as well as various data synthesis methods. In this section, we extend this analysis by studying if the synthetic finetuning is still useful with M2M initialization, and if it is universally helpful across different languages. We also examine the label balancing factor $\mu$ and take a detailed look at the prediction errors.

**Synthetic Finetuning**  We redo the analysis on en-de synthetic finetuning with the smaller 2M parallel sentence samples from Europarl, as in the WMT 2020 experiments, but with the updated test20-v2 test set and models with M2M-100-small initialization. The results (shown in Table 6.7) largely corroborate the findings of the WMT 2020 experiments, showing that src-mt1-mt2 and mvppe are the two most helpful data synthesis methods. We then extend those two methods to ro-en and et-en, using the up-to-date Bing Translator production model as the stronger MT system (a.k.a. mt2) in the src-mt1-mt2 synthetic data. The result (shown in Table 6.8) is mixed, with mvppe failing to improve performance for both language pairs, and src-mt1-mt2 only being helpful for et-en language pair. We also trained two extra ro-en and et-en LevT models using the respective Bing Translator models to generate the KD data, which neither help improve performance on their own nor work better with src-mt1-mt2 synthetic data.
CHAPTER 6. IMPROVING QUALITY ESTIMATION MODELS THROUGH LEVENSHTEIN TRAINING

We notice that the mvppe synthetic data seems to significantly improve the F1 score of the OK label in general, for which we don’t have a good explanation.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Target MCC</th>
<th>F1-OK</th>
<th>F1-BAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>ro-en $\mu = 1.0$</td>
<td>0.612</td>
<td>0.949</td>
<td>0.659</td>
</tr>
<tr>
<td>ro-en $\mu = 3.0$</td>
<td>0.577</td>
<td>0.930</td>
<td>0.619</td>
</tr>
<tr>
<td>ru-en $\mu = 1.0$</td>
<td>0.267</td>
<td><strong>0.960</strong></td>
<td>0.284</td>
</tr>
<tr>
<td>ru-en $\mu = 3.0$</td>
<td><strong>0.339</strong></td>
<td>0.943</td>
<td><strong>0.390</strong></td>
</tr>
<tr>
<td>et-en $\mu = 1.0$</td>
<td>0.478</td>
<td><strong>0.933</strong></td>
<td>0.511</td>
</tr>
<tr>
<td>et-en $\mu = 3.0$</td>
<td><strong>0.512</strong></td>
<td>0.925</td>
<td><strong>0.587</strong></td>
</tr>
<tr>
<td>ne-en $\mu = 1.0$</td>
<td>0.660</td>
<td><strong>0.885</strong></td>
<td>0.774</td>
</tr>
<tr>
<td>ne-en $\mu = 3.0$</td>
<td><strong>0.681</strong></td>
<td>0.855</td>
<td><strong>0.788</strong></td>
</tr>
</tbody>
</table>

Table 6.9: Analysis of different label balancing factors initialized on to-English language pairs. All results are based on the multilingual model and not performing synthetic finetuning step.

**Label Balancing Factor** We find the quality estimation task performance to be quite sensitive to the label balancing factor $\mu$, but there is also no universally optimal value for all language pairs. Table 6.9 shows this behavior for all to-English language pairs. Notice that while for most of the cases $\mu$ simply controls a trade-off between the performance of OK and BAD outputs, there are also cases such as ro-en where a certain choice of $\mu$ hurts the performance of both classes. This might be due to a certain label class being particularly hard to fit, thus creating more difficulties with learning when the loss function is designed to skew to this label class.

It should be noted that this label balancing factor does not correlate directly with the
CHAPTER 6. IMPROVING QUALITY ESTIMATION MODELS THROUGH LEVENSHEIN TRAINING

Table 6.10: Detailed evaluation metric breakdown of all submitted ensemble system on test20 test set.

<table>
<thead>
<tr>
<th>Lang.</th>
<th>Tgt. MCC</th>
<th>MT MCC</th>
<th>MT BAD (P/R/F1)</th>
<th>MT OK (P/R/F1)</th>
<th>GAP MCC</th>
<th>GAP BAD (P/R/F1)</th>
<th>GAP OK (P/R/F1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>en-de</td>
<td>0.504</td>
<td>0.503</td>
<td>0.476</td>
<td>0.731</td>
<td>0.576</td>
<td>0.950</td>
<td>0.863</td>
</tr>
<tr>
<td>en-zh</td>
<td>0.466</td>
<td>0.381</td>
<td>0.467</td>
<td>0.787</td>
<td>0.586</td>
<td>0.879</td>
<td>0.633</td>
</tr>
<tr>
<td>ro-en</td>
<td>0.612</td>
<td>0.645</td>
<td>0.729</td>
<td>0.709</td>
<td>0.719</td>
<td>0.922</td>
<td>0.929</td>
</tr>
<tr>
<td>ru-en</td>
<td>0.349</td>
<td>0.329</td>
<td>0.296</td>
<td>0.675</td>
<td>0.411</td>
<td>0.945</td>
<td>0.775</td>
</tr>
<tr>
<td>et-en</td>
<td>0.575</td>
<td>0.555</td>
<td>0.678</td>
<td>0.681</td>
<td>0.679</td>
<td>0.875</td>
<td>0.873</td>
</tr>
<tr>
<td>ne-en</td>
<td>0.694</td>
<td>0.434</td>
<td>0.760</td>
<td>0.918</td>
<td>0.832</td>
<td>0.746</td>
<td>0.454</td>
</tr>
</tbody>
</table>

The ratio of the OK vs. BAD labels in the training set. For example, to obtain the best performance, ne-en requires $\mu = 3.0$ while en-de requires $\mu = 1.0$, while the OK to BAD ratio for ne-en (2.14:1) is much less skewed compare to en-de (10.2:1).

**Detailed Error Breakdown** We found it hard to develop an intuition for the model performance from the MCC metric. To further understand which label categories our models struggle with the most, we breakdown the target-side metric into a cross product of \{MT, GAP\} tags and \{OK, BAD\} classes and compute precision, recall and F1-score for each category. The breakdown is shown in Table 6.10. It can be seen that our model is making the most mistakes with the GAP BAD category, while the category with the least mistakes is the GAP OK category. Also, note that for MT word tags, the models often seem to suffer more from low precision rather than low recall, while for gaps it is the opposite.

Overall, we see that the highest F1 scores we can achieve for detecting bad MT words or gaps are rarely higher than 0.8, which indicates that there should be ample room for improvement. It would also be interesting to measure the inter-annotator agreement of these
CHAPTER 6. IMPROVING QUALITY ESTIMATION MODELS THROUGH LEVENSHTEIN TRAINING

word-level quality labels, in order to get a sense of the level of human performance we should be aiming for.

6.6 Conclusion

In this chapter, we proposed to improve the existing dedicated quality estimation model by using Levenshtein Transformer to substitute the usual MLM-style training in the Predictor-Estimator framework as the initial training step. We also proposed a series of techniques to effectively transfer the translation knowledge to the word-level QE task, including data synthesis, heuristic subword-level reference, and incorporating pre-trained translation models. On WMT 2020 and WMT 2021 datasets, our experiment results demonstrate superior data efficiency under the data-constrained setting and competitive performance under the unconstrained setting. We also hope this work can inspire further exploration for other uses of Levenshtein Transformer apart from the non-autoregressive neural machine translation.
Chapter 7

Conclusion

7.1 Summary of Findings

In this thesis, we have looked at several facets of the runtime model audit problem in the context of sequence models for natural language processing. We now summarize some of our key findings in this thesis:

- In Chapter 3, by evaluating the plausibility and faithfulness of model explanations in the context of various neural language model architectures, we have shown that model explanations can fail by either having poor plausibility or faithfulness, and that the quality of explanation is highly variable with the choice of model architecture, model configuration, and explanation method. We also show that distilled models
CHAPTER 7. CONCLUSION

tend to be more compatible with gradient-based post-hoc explanation methods, thus pointing out a potential path to improve explainability while maintaining the model performance.

• In Chapter 4, by comparing the quality of explanation with dedicated models that are designed to induce word alignments with methods that give explanations of neural machine translation models in the same format, we have shown that for the quality of explanations for neural machine translation models has reached a level that is on-par with several widely adopted word alignment models.

• In both Chapter 3 and 4, in the context of two different applications, we have shown that composition scheme is an important aspect of applying post-hoc interpretation method to NLP that should not be overlooked. Among the two composition schemes we have compared, Gradient · Input is better than the Vector Norm composition scheme in almost all of our evaluations.

• In Chapter 5, by comparing the correlation between human post-edits and the word-level quality labels, we have shown that the unsupervised quality estimation method based on the posterior probability from translation decoding has not reached a comparable performance as the standard Predictor-Estimator approach for estimating output quality on the word-level.
In Chapter 6, we have shown that we can advance the state of the art created by Predictor-Estimator models, by starting with a non-autoregressive machine translation architecture, Levenshtein Transformer. Essentially, we argue that the special training procedure for Levenshtein Transformer is a pre-training scheme that is more targeted to the end task we would like to tackle. Besides, we have also shown that aspects like proper conversion between word-level and subword-level labels, synthetic finetuning, and diverse ensembles play a role in building an optimal word-level quality estimation system.

7.2 Future Work

Our work is not without its specific contexts, and there are areas beyond such contexts where further explorations are warranted. We will now motivate work in some of these areas with methodologies or observations from this thesis.

Generalizability of Findings and Evaluations

Our study sits in the context of sequence models, and the context of two different applications, namely language modeling, and machine translation. Beyond this context, while many of our proposals take application specifications into account, there are contributions to
CHAPTER 7. CONCLUSION

model explanations that are generalizable beyond such contexts, for example, our proposed
twofold evaluation scheme (plausibility/faithfulness), and the Gradient · Input composition
scheme. On the other hand, we also observed a lot of variations in the behaviors of model
explanation methods across different tasks. For example, we observed high-quality explana-
tions for convolutional sequence-to-sequence models for the machine translation task, but
noisy explanations for QRNN language models, which is also a convolutional architecture.
Future explorations could further quantify the generalizability of our findings by training the
same model with loss functions for different applications and systematically investigating
the interaction between the task-oriented objective and the audit method performance.

As we have pointed out in Chapter 3, the evaluation datasets for model explanations
still need to be created in an application-specific manner. The same applies to quality
estimation – while research on quality estimation also exists for other applications (such as
grammatical error correction (Napoles, Sakaguchi, and Tetreault, 2016) and summarization
(Xenouleas et al., 2019)), there hasn’t been a systematic cross-application evaluation yet.
We think the community should extend existing evaluations to more applications and seek
general evaluations that can be used for cross-application comparisons. Moreover, during
our literature review, we realized that there is a significant skew in the model explanation
literature towards classification tasks (as opposed to sequence generation tasks), which
is partly exemplified in the form of data and resources in this survey by Wiegreffe and
Marasovic (2021). Our study helped ease this skew, and we call for more work on model explanations in the context of sequence modeling, especially given the immense modeling power of large pre-trained generative language models such as GPT-3 (Brown et al., 2020) and OPT (Zhang et al., 2022).

**Types of Audit Signals**

In this thesis, we studied two types of auxiliary signals for auditing – signals that explain model predictions and signals that estimate the qualities of the model outputs. Our approach largely treats these two signals as disjoint, but some studies combined these signals and obtained interesting results. For example, Treviso et al. (2021) formulates the problem of word-level quality estimation as explaining the prediction from a sentence-level quality estimation system. Besides, the feature-rich quality estimation models have long adopted word alignments as a useful input feature (e.g., Martins, Kepler, and Monteiro (2017)). It would be interesting to see if our findings on these different types of audit signals could potentially be combined to further advance the state-of-the-art.

Our discussions focus on generating some existing auxiliary signals but do not attempt to build any novel ones. Exploration of novel audit signals based on user studies is an important topic, especially since some recent studies pointed out that users’ preference for audit signals may not necessarily align with the researcher’s original intuition for design
CHAPTER 7. CONCLUSION

(Shen et al., 2022). It should be re-iterated that any design for auxiliary signals should always keep the user’s goal and application context in mind, and should not cause cognitive overload for users.

Audit Methods

Both Chapter 3 and Chapter 5 have pointed out that choosing the appropriate method plays a major role in extracting useful auxiliary signals for runtime model auditing. However, we also chose to concentrate on gradient-based methods when studying post-hoc explanation methods based on the original inference model. This choice was driven by the fact that we would like to study model auditing with various state-of-the-art models and architectures and under reasonably realistic use-cases for respective applications. Hence, it is not feasible for us to manually re-define back-propagation rules to study pseudo-gradient-based methods such as LRP and DeepLIFT, or enumerate a large input space to study perturbation-based methods such as Shapley value. It would be interesting to see if re-formulations of pseudo-gradient-based methods (introduced by Ancona et al. (2018)) that are more compatible with modern deep learning toolkits will work better for our model explanation setup.
CHAPTER 7. CONCLUSION

Human Study

The ultimate goal of our research is to help humans more confidently use AI and reduce adverse effects. As with most NLP studies, our study relies on human knowledge extracted through the data annotation process as a proxy to measure our progress on this goal – for model explanations, the human intuitions about why certain predictions are made; for quality estimations, the human judgment as to whether edits are required for individual word predictions. While direct human assessment of the auxiliary signals may sound enticing, they are not straightforward to deploy in practice at times due to implicit human cognition bias (Herman, 2017). For example, a faithful explanation for an erroneous prediction from a model may be perceived as implausible for someone who does not have much knowledge about AI systems and relies on their decision criteria to perform evaluations. This was also one of the reasons why we chose to leverage existing linguistic annotations for plausibility evaluations of model explanations in Chapter 3.

In the future, we think a pragmatic evaluation through human-computer collaboration could be a promising way to evaluate the quality of auxiliary signals. This approach was originally proposed by Feng and Boyd-Graber (2019) for evaluating model explanations of a question-answering model. A similar approach could be adapted to a machine translation setting by asking a human who does not know a language to estimate the quality of a translation by only leveraging the auxiliary signals such as word alignments and word-
CHAPTER 7. CONCLUSION

level quality labels, and comparing their estimation with direct assessment from bilingual speakers.

**Improving Output Quality with Audit Signals**

Apart from helping humans, there are also prospects of using model audit signals to improve the model output quality. Some obvious examples are real-world tasks where humans collaborate with sequence generation models to accomplish a task, such as translation post-editing, computer-assisted writing, and computer-aided translation. Besides, some recent studies also attempt to complete the feedback loop by integrating audit signals into the training/inference process, e.g., removing noisy features with explanations (Bento et al., 2021) and improving search with quality estimation (Fernandes et al., 2022). In the long run, we think this is a promising path that could lead to both academic advances in these methods as well as improved result delivery for sequence generation tasks.
Appendix A

Appendix for Chapter 3

A.1 Data Filtering Details

A.1.1 Penn Treebank (PTB)

A potential candidate for a test case is extracted every time a word with POS tag VBZ (Verb, 3rd person singular present) or VBP (Verb, non-3rd person singular present), or a copula that is among *is, are, was, were*, shows up. The candidate will then be filtered subjecting to the following criteria:

1. The prefix has at least one attractor word (a noun that has a different morphological number as the verb that is predicted). This is to ensure that evaluation could be
APPENDIX A. APPENDIX FOR CHAPTER 3

conducted in the alternative scenario.

2. The verb cannot immediately follow its grammatical subject (note: it may still immediately follow a cue word that is not a grammatical subject). This is to ensure that the signal of the subject is not overwhelmingly strong compared to the attractors.

3. Not all attractors occur earlier 10 words than the grammatical subject. Same reason as the previous criteria.

Overall, we obtained 1448 test cases out of 49168 sentences in PTB (including train, dev, and test set). We lose a vast majority of sentences mostly because of the last two criteria.

A.1.2 Syneval

We use the following sections of the original data (followed by their names in the data dump, Marvin and Linzen, 2018):

- Agreement in a sentential complement: sent_comp
- Agreement across a prepositional phrase: prep.anim and prep.inanim
- Agreement across a subject relative clause: subj.rel
- Agreement across an object relative clause:
  - obj.rel.across.anim, obj.rel.across.inanim, obj.rel.no.comp.across.anim,
APPENDIX A. APPENDIX FOR CHAPTER 3

\texttt{obj\_rel\_no\_comp\_across\_inanim}

- Agreement within an object relative clause:
  \texttt{obj\_rel\_within\_anim, obj\_rel\_within\_inanim, obj\_rel\_no\_comp\_within\_anim,}
  \texttt{obj\_rel\_no\_comp\_within\_inanim}

We select these sections because they all have strong interfering attractors or have cues that may potentially be mistaken as attractors. We obtained much fewer examples (6280) than the original data (249760) because lots of examples only differ in the verb or the object they use, which become duplicates when we extract prefix before the verb.

The original dataset does not come with cue/attractor annotations, but it can be easily inferred because they are generated by simple heuristics.

Note that most of these sections have only around 50% prediction accuracy with RNNs in the original paper. Our results on large-scale language models corroborate the findings in the original paper.

A.1.3 CoNLL

We use the dataset (Pradhan et al., 2012) with gold parses, entities mentions, and mention boundaries. A potential candidate for a test case is extracted every time a pronoun shows up. The male pronouns are \textit{he}, \textit{him}, \textit{himself}, \textit{his}, while the female pronouns are \textit{she}, \textit{her},
APPENDIX A. APPENDIX FOR CHAPTER 3

*herself, hers.* We don’t include cases of epicene pronouns like *it, they,* etc. because they often involve tricky cases like entity mentions covering a whole clause. We break prefixes according to the document boundaries as provided in the original dataset unless the prefix is longer than 512 words, in which case we instead break at the nearest sentence boundary.

The annotation for this dataset does not cover the gender of entities. We are aware that the original shared task provides gender annotation, but to this day, the documentation for the data is missing and hence we cannot make use of this annotation. Hence, we instead used several heuristics to infer the gender of an entity mention, in descending order:

- If an entity mention and a pronoun have a coreference relationship, they should share the same gender.

- If an entity mention starts with “Mr.” or “Mrs.” or “Ms.”, we assign the corresponding gender.

- If the entity mention has a length of two tokens, we assume it’s a name and use gender inference tools¹ to guess its gender. Note that the gender guesser may also indicate that it’s not able to infer the gender, in that case, we do not assign a gender.

- If a mention is co-referenced with another mention that is not a pronoun, they should also have the same gender.

¹https://github.com/lead-ratings/gender-guesser

184
Manual inspection of the resulting data indicates that the scheme above covers the gender of most entity mentions correctly. We hope that our dataset could be further perfected by utilizing higher quality annotation on entity genders.

Since each entity mention could span more than one word, we add all words within the span into their corresponding cue/attractor set. A tricky case is where two entity mention spans are nested or intersected. For the first case, we exclude a smaller span from the larger one to create two unintersected spans as the new span for the cue/attractor set. For the second case, we exclude the intersecting parts from both spans.

Finally, all candidates are filtered subject to the following two criteria:

1. The prefix should include one attractor entity.

2. The entity mention that is closest to the verb should be of different gender (either the opposite or epicene).

We obtained 586 document segments from the 2280 documents in the original data. As pointed out in Zhao et al. (2018), the CoNLL dataset is significantly biased towards male entity mentions. Nevertheless, our filtering scheme generated a relatively balanced test set: among 586 test cases, 258 are male pronouns, while 328 are female pronouns.
A.1.4 Winobias

We used the same data as the unambiguous coreference resolution dataset in Jumelet, Zuidema, and Hupkes (2019), which is in turn generated by a script from Zhao et al. (2018), except that we excluded cases where both nouns in the sentence are of the same gender. Similar to Syneval dataset, the cue and attractors could easily be inferred with heuristics.

A.2 Language Model Perplexities

Parameter size and perplexity on WikiText-103 dev set for all language models are shown in Table A.1 for reference.

Below are the respective commands to reproduce these results.

- LSTM: python -u main.py -epochs 50 -nlayers 3 -emsize 400 -nhid 2000 -dropoute 0 -dropouth 0.01 -dropouti 0.01 -dropout 0.4 -wdrop 0.2 -bptt 140 -batch_size 60 -optimizer adam -lr 1e-3 -data data/wikitext-103 -save save -when 25 35 -model LSTM

- QRNN: python -u main.py -epochs 14 -nlayers 4 -emsize 400 -nhid 2500 -alpha 0 -beta 0 -dropoute 0 -dropouth 0.1 -dropouti 0.1 -dropout 0.1 -wdrop 0 -wdecay 0 -bptt 140 -batch_size 40 -optimizer adam -lr 1e-3 -data data/wikitext-103 -save save -when 12 -model QRNN
• Transformer: python train.py -task language_modeling data-bin/wikitext-103
  -save-dir checkpoints -arch transformer_lm_wiki103 -decoder-layers $layers
  -decoder-attention-heads $num_heads -decoder-embed-dim $emb
  -decoder-ffn-embed-dim $width -max-update 286000 -max-lr 1.0
  -t-mult 2 -lr-period-updates 270000 -lr-scheduler cosine -lr-shrink 0.75
  -warmup-updates 16000 -warmup-init-lr 1e-07 -min-lr 1e-09
  -optimizer nag -lr 0.0001 -clip-norm 0.1 -criterion adaptive_loss
  -max-tokens 3072 -update-freq 3 -tokens-per-sample 3072 -seed 1
  -sample-break-mode none -skip-invalid-size-inputs-valid-test -ddp-backend=no_c10d

A.3 Additional Interpretation Examples

We show some additional interpretations generated by the state-of-the-art LSTM (Table A.2), QRNN (Table A.3) and Transformer (Table A.4) models on PTB and CoNLL dataset, with their respective best-performing interpretation method.
### Table A.1: Parameter size (in millions) and perplexity on WikiText-103 dev set for all language models we trained.

<table>
<thead>
<tr>
<th>Architectures</th>
<th>Layers</th>
<th>Config</th>
<th>Params (M)</th>
<th>dev ppl</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM</td>
<td>3</td>
<td>-</td>
<td>162</td>
<td>37.65</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>-</td>
<td>130</td>
<td>41.97</td>
</tr>
<tr>
<td>QRNN</td>
<td>4</td>
<td>-</td>
<td>154</td>
<td>32.12</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>-</td>
<td>135</td>
<td>36.54</td>
</tr>
<tr>
<td>Transformer</td>
<td>16</td>
<td>4096w_1024e_8h</td>
<td>247</td>
<td>17.97</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>4096w_1024e_8h (Distill Student)</td>
<td>96.1</td>
<td>24.92</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>4096w_1024e_8h</td>
<td>96.1</td>
<td>28.96</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>1024w_512e_2h</td>
<td>39.3</td>
<td>32.63</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>1024w_512e_4h</td>
<td>39.3</td>
<td>32.09</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>1024w_512e_8h</td>
<td>39.3</td>
<td>31.38</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>512w_512e_8h</td>
<td>35.1</td>
<td>33.99</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>2048w_512e_8h</td>
<td>47.7</td>
<td>30.19</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>1024w_256e_8h</td>
<td>17.4</td>
<td>41.56</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>1024w_1024e_8h</td>
<td>96.1</td>
<td>27.01</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1024w_512e_2h</td>
<td>30.8</td>
<td>37.03</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1024w_512e_4h</td>
<td>30.8</td>
<td>35.67</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1024w_512e_8h</td>
<td>30.8</td>
<td>35.82</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>512w_512e_8h</td>
<td>28.7</td>
<td>38.34</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>2048w_512e_8h</td>
<td>35.0</td>
<td>33.70</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1024w_256e_8h</td>
<td>14.3</td>
<td>48.47</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1024w_1024e_8h</td>
<td>70.9</td>
<td>30.46</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1024w_512e_2h</td>
<td>26.6</td>
<td>44.45</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1024w_512e_4h</td>
<td>26.6</td>
<td>42.23</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1024w_512e_8h</td>
<td>26.6</td>
<td>41.86</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>512w_512e_8h</td>
<td>25.6</td>
<td>44.97</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2048w_512e_8h</td>
<td>28.7</td>
<td>38.99</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1024w_256e_8h</td>
<td>12.7</td>
<td>59.16</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1024w_1024e_8h</td>
<td>58.3</td>
<td>36.06</td>
</tr>
</tbody>
</table>
PTB
1- (U.S.) (Trade) (Representative) (Carla) (Hills) said the first dispute-settlement (panel) set up under the U.S.-Canadian “free (trade)” (agreement) has ruled that (Canada) ’s [restrictions] on [exports] of (Pacific) (salmon) and (herring) | PLURAL
2- Individual [investors], (investment) [firms] and [arbitragers] who speculate in the [stocks] of (takeover) [candidates] can suffer (liquidity) and (payment) [problems] when [stocks] dive ; those [investors] often | PLURAL
3- (U.S.) [companies] wanting to expand in [Europe] | PLURAL
4- CURBING [WAGE] (BOOSTS) will get high [priority] again in 1990 collective [bargaining] , a [Bureau] of [National] [Affairs] [survey] of 250 (companies) with (pacts) expiring next [year] | PLURAL
5- TEMPORARY (WORKERS) have good (educations) , the [National] [Association] of [Temporary] [Services] | SINGULAR

CoNLL
1- [Israeli] [Prime] [Minister] [Ehud] [Barak] says [he] is freezing tens of millions of dollars in tax payments to the Palestinian Authority . [Mr.] [Barak] says [he] is withholding the money until the Palestinians abide by cease - fire agreements . Earlier Thursday [Mr.] [Barak] ruled out an early resumption of peace talks , even with the United States acting as intermediary . (Eve) (Conette) reports from Jerusalem as Defending what | MALE
2- Once again there ’ll be two presidential candidates missing from the debate . Pat Buchanan hardly registers on the political radar this year . And Ralph Nader , who may make the difference between a [Gore] or [Bush] win in several places . (ABC) (’s) (Linda) (Douglas) was with | FEMALE

Table A.2: Addition interpretation examples with LSTM.
Table A.3: Addition interpretation examples with QRNN.
PTB
1- (U.S.) (Trade) (Representative) (Carla) (Hills) said the first dispute-settlement (panel) set up under the U.S.-Canadian “free (trade)” (agreement) has ruled that (Canada)’s [restrictions] on [exports] of (Pacific) (salmon) and (herring) | PLURAL
2- Individual [investors], (investment) [firms] and [arbitrages] who speculate in the [stocks] of (takeover) [candidates] can suffer (liquidity) and (payment) [problems] when [stocks] dive; those [investors] often | PLURAL
3- (U.S.) [companies] wanting to expand in (Europe) | PLURAL
4- CURBING [WAGE] (BOOSTS) will get high [priority] again in 1990 collective [bargaining], a [Bureau] of [National] [Affairs] [survey] of 250 (companies) with (pacts) expiring next [year] | PLURAL
5- TEMPORARY (WORKERS) have good (educations) in the [National] [Association] of [Temporary] [Services] | SINGULAR

CoNLL
1- [Israeli] [Prime] [Minister] [Ehud] [Barak] says [he] is freezing tens of millions of dollars in tax payments to the Palestinian Authority. [Mr.] [Barak] says [he] is withholding the money until the Palestinians abide by cease-fire agreements. Earlier Thursday [Mr.] [Barak] ruled out an early resumption of peace talks, even with the United States acting as intermediary. [Eve] (Conette) reports from Jerusalem. Defending what | FEMALE
2- Once again there’ll be two presidential candidates missing from the debate. Pat Buchanan hardly registers on the political radar this year. And Ralph Nader, who may make the difference between a [Gore] or [Bush] win in several places | (ABC) (‘s) (Linda) (Douglas) was with | MALE

Table A.4: Addition interpretation examples with Transformer.
Bibliography


BIBLIOGRAPHY


Bastings, Jasmijn and Katja Filippova (Nov. 2020). “The elephant in the interpretability room: Why use attention as explanation when we have saliency methods?” In: Proceedings
BIBLIOGRAPHY


BIBLIOGRAPHY


Bouma, Gosse and Yannick Parmentier, eds. (2014). Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2014,
BIBLIOGRAPHY


URL: http://aclweb.org/anthology/E/E14/.


BRIEBIOGRAPHY


Brier, Glenn W et al. (1950). “Verification of forecasts expressed in terms of probability”. In: Monthly weather review 78.1, pp. 1–3.


BIBLIOGRAPHY

Hsuan-Tien Lin. url: https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfc4967418bfb8ac142f64a-Abstract.html.


BIBLIOGRAPHY


Cheng, Yong, Lu Jiang, Wolfgang Macherey, and Jacob Eisenstein (July 2020b). “AdvAug: Robust Adversarial Augmentation for Neural Machine Translation”. In: Pro-

Clark, Kevin, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning (2020). “ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators”. In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. url: https://openreview.net/forum?id=r1xMH1BtvB.


BIBLIOGRAPHY


Dong, Li, Chris Quirk, and Mirella Lapata (July 2018). “Confidence Modeling for Neural Semantic Parsing”. In: Proceedings of the 56th Annual Meeting of the Association for
BIBLIOGRAPHY


Fan, Angela, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, Naman Goyal, Tom Birch, Vitaliy Liptchinsky, Sergey Edunov, Edouard Grave, Michael Auli,


BIBLIOGRAPHY


BIBLIOGRAPHY


Hassan, Hany, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, Shujie


Heider, Fritz (2013). The psychology of interpersonal relations. Psychology Press.

BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


Khayrallah, Huda and Philipp Koehn (July 2018). “On the Impact of Various Types of Noise on Neural Machine Translation”. In: Proceedings of the 2nd Workshop on Neural Ma-
BIBLIOGRAPHY


BIBLIOGRAPHY


229


BIBLIOGRAPHY


BIBLIOGRAPHY


Li, Jiwei, Xinlei Chen, Eduard Hovy, and Dan Jurafsky (June 2016). “Visualizing and Understanding Neural Models in NLP”. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Lan-


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


See, Abigail, Peter J. Liu, and Christopher D. Manning (July 2017). “Get To The Point: Summarization with Pointer-Generator Networks”. In: Proceedings of the 55th Annual
BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


Treviso, Marcos, Nuno M. Guerreiro, Ricardo Rei, and André F. T. Martins (Nov. 2021). “IST-Unbabel 2021 Submission for the Explainable Quality Estimation Shared Task”. In:


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


Zhu, Jinhua, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wengang Zhou, Houqiang Li, and Tie-Yan Liu (2020). “Incorporating BERT into Neural Machine Translation”. In: 8th Inter-
Vita

Shuoyang Ding is a Ph.D. candidate in the Center for Language and Speech Processing (CLSP), Johns Hopkins University (JHU) advised by Professor Philipp Koehn. His research lies at the crossroad of machine translation, natural language processing, and Explainable AI (XAI), with a particular interest in helping humans to understand and interact with natural language generation systems such as language models and machine translation models. Apart from that, he has also done some work on syntactic parsing and speech recognition, the latter of which won the best student paper award at Interspeech 2018. Outside JHU, Shuoyang previously interned at Amazon, Salesforce, Microsoft, and conducted research at the University of Edinburgh. Shuoyang holds a Bachelor’s degree from Beijing University of Posts & Telecommunications (BUPT).