

HARVARD UNIVERSITY  
Graduate School of Arts and Sciences



DISSERTATION ACCEPTANCE CERTIFICATE

The undersigned, appointed by the

Harvard John A. Paulson School of Engineering and Applied Sciences  
have examined a dissertation entitled:

“Human-AI Collaboration for Natural Language Generation with  
Interpretable Neural Networks”

presented by: Sebastian Gehrman

candidate for the degree of Doctor of Philosophy and here by  
certify that it is worthy of acceptance.

Signature

Typed name: Professor B. Grosz

Signature

Typed name: Professor S. Rush

Signature

Typed name: Professor S. Shieber

October 18, 2019



# Human-AI Collaboration for Natural Language Generation with Interpretable Neural Networks

A DISSERTATION PRESENTED

BY

SEBASTIAN GEHRMANN

TO

THE JOHN A. PAULSON SCHOOL OF ENGINEERING AND APPLIED SCIENCES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN THE SUBJECT OF

COMPUTER SCIENCE

HARVARD UNIVERSITY

CAMBRIDGE, MASSACHUSETTS

MAY 2020

©2019 – SEBASTIAN GEHRMANN  
CREATIVE COMMONS ATTRIBUTION LICENSE 4.0.  
YOU ARE FREE TO SHARE AND ADAPT THESE MATERIALS FOR ANY PURPOSE  
IF YOU GIVE APPROPRIATE CREDIT AND INDICATE CHANGES.

# Human-AI Collaboration for Natural Language Generation with Interpretable Neural Networks

## ABSTRACT

Using computers to generate natural language from information (NLG) requires approaches that plan the content and structure of the text and actualize it in fluent and error-free language. The typical approaches to NLG are data-driven, which means that they aim to solve the problem by learning from annotated data. Deep learning, a class of machine learning models based on neural networks, has become the standard data-driven NLG approach. While deep learning approaches lead to increased performance, they replicate undesired biases from the training data and make inexplicable mistakes. As a result, the outputs of deep learning NLG models cannot be trusted. We thus need to develop ways in which humans can provide oversight over model outputs and retain their agency over an otherwise automated writing task.

This dissertation argues that to retain agency over deep learning NLG models, we need to design them as team members instead of autonomous agents. We can achieve these team member models by considering the interaction design as an integral part of the machine learning model development. We identify two necessary conditions of team member-models – interpretability and controllability. The models need to follow a reasoning process such that human team members can understand and comprehend it. Then, if humans do not agree with the model, they should be able to change the reasoning process, and the model should adjust its output accordingly.

In the first part of the dissertation, we present three case studies that demonstrate how interactive interfaces can positively affect how humans understand model predictions. In the second part, we introduce a neural network-based approach to document summarization that directly models the selection of relevant content. We show that, through this selection, a human user can control what part of a document the algorithm summarizes. In the final part of this dissertation, we show that this design approach, coupled with an interface that exposes these interactions, can lead to a forth and back between human and autonomous agents where the two actors collaboratively generate text. This dissertation thus demonstrates how to develop models with these properties and how to design neural networks as team members instead of autonomous agents.

# Contents

<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
1.1	Thesis Overview . . . . .	7
1.2	Contributions . . . . .	11
<b>2</b>	<b>NATURAL LANGUAGE GENERATION</b>	<b>13</b>
2.1	Notation . . . . .	15
2.2	Natural Language Generation Tasks . . . . .	15
2.3	Deep Learning for Natural Language Processing . . . . .	20
2.4	Approximate Search and Text Generation . . . . .	34
<b>3</b>	<b>UNDERSTANDING USERS AND THEIR INTERPRETABILITY NEEDS</b>	<b>36</b>
3.1	User Types: Architects, Trainers, and End Users . . . . .	42
3.2	Design Space for Integrating Machine Learning into Interfaces . . . . .	44
<b>4</b>	<b>EVALUATING EXPLANATIONS IN AUTOMATED PATIENT PHENOTYPING</b>	<b>51</b>
4.1	Phenotyping in large EHR datasets . . . . .	54
4.2	Data . . . . .	56
4.3	Methods . . . . .	57
4.4	Deriving Salient Features . . . . .	60
4.5	Evaluation . . . . .	62
4.6	Results . . . . .	64
4.7	Discussion and Conclusion . . . . .	68
<b>5</b>	<b>INTERACTIVELY UNDERSTANDING RECURRENT NEURAL NETWORKS</b>	<b>70</b>
5.1	Visualization for understanding neural networks . . . . .	72
5.2	User Analysis and Goals . . . . .	73
5.3	Design of LSTMVis . . . . .	75
5.4	Use Cases . . . . .	81
5.5	Long-Term Case Study . . . . .	86
5.6	Conclusion . . . . .	88

<b>6</b>	<b>DEBUGGING PREDICTIONS OF SEQUENCE-TO-SEQUENCE MODELS</b>	<b>90</b>
6.1	Motivating Case Study: Debugging Translation . . . . .	94
6.2	Goals and Tasks . . . . .	98
6.3	Design of SEQ2SEQ-VIS . . . . .	101
6.4	Use Cases . . . . .	106
6.5	Conclusions . . . . .	111
<b>7</b>	<b>BOTTOM-UP SUMMARIZATION: EXTENDING MODELS WITH CONTROLLABLE VARIABLES</b>	<b>113</b>
7.1	Related Work . . . . .	116
7.2	Background: Neural Summarization . . . . .	118
7.3	Bottom-Up Attention . . . . .	119
7.4	Inference . . . . .	123
7.5	Data and Experiments . . . . .	124
7.6	Results . . . . .	125
7.7	Analysis and Discussion . . . . .	128
7.8	Conclusion . . . . .	132
<b>8</b>	<b>COLLABORATIVE SEMANTIC INFERENCE</b>	<b>134</b>
8.1	Interactive Collaboration . . . . .	136
8.2	Rearchitecting models to enable collaborative semantic inference . . . . .	139
8.3	Use Case: A Collaborative Summarization Model . . . . .	143
8.4	Details on the Summarization Model Hooks . . . . .	145
8.5	Towards a Co-Design Process for CSI Systems . . . . .	154
8.6	Conclusions . . . . .	156
<b>9</b>	<b>DISCUSSION AND CONCLUSION</b>	<b>158</b>
9.1	The evaluation of text-generating models . . . . .	159
9.2	The ethical permissibility of text-generating models . . . . .	162
9.3	Collaboration in other domains . . . . .	166
9.4	Conclusion . . . . .	167
	<b>REFERENCES</b>	<b>169</b>

# Acknowledgments

In graduate school, you are thrown into the deep water and your advisors teach you to swim as you go along. The first time I wrote a paper, my advisors told me that they would like me to write it again. For my latest paper, I received small editorial comments. I am, therefore, confident that I have learned to swim. For that, I would like to thank my advisors Barbara Grosz and Sasha Rush, who have given me the opportunity to become a researcher and have supported me in so many ways.

Barbara was first person who exposed me to AI, and I am forever grateful for taking me taking me on as a student and believing in me to grow as a researcher. I know most things I know about NLP thanks for Sasha's guidance who imprinted on me his striving for flawlessly presented arguments and well-crafted code. Their support has been unwavering and their advice and subtle nudges continue to steer me in the right direction.

The list of people who helped make this dissertation possible continues with Stuart Shieber, who never fails to amaze with his thoughtful suggestions and impromptu lectures on his whiteboard. I am grateful to Krzysztof Gajos for accepting me as a perpetual guest into the HCI group. Our conversations indirectly influenced much of this dissertation. I will miss our 11am tea's and participating in paper trashing sessions. Yet another person who I am happy to count as an advisor and friend is Ofra Amir. She guided my first steps in the research world, first as advisor and later as an office mate. Her advice is always useful, be it about AI or hiking.

Despite my great advisors, my research would be nothing without the support from my collaborators. This list cannot start without thanking Hendrik Strobelt, who over the last years has been instrumental in most work presented in this dissertation. Despite our different backgrounds, we somehow learned each others language and managed to successfully work on projects. Discussions are always better over cake. Continuing the list of amazing collaborators, I would further like to thank David Grant, Eric Carlson, Falcon Dai, Franck Dernoncourt, Hanspeter Pfister, Henry Elder, Jody Lin, Joy Wu, Lauren Urke, Lee Sanders, Leo Celi, Michael Behrisch, Mirac Suzgun, Ned Moseley, Patrick Tyler, Payel Das, Robert Krüger, Steven Layne, Tom Sercu, Yeran Li, Yonatan Belinkov, Yun-tian Deng, and Zach Ziegler for all their contributions toward making this dissertation possible.

I am also extremely thankful for all the great experiences I had during my internships. I got to work with a group of wonderful people at the Bruno Kessler Foundation in Italy, most of all Alessandro Cappelletti, Eleonora Mencarini, Gianluca Shiavo, Massimo Zancanaro, and Oliviero Stock. Similarly, I happily look back at my times at Adobe where I got to work with Carl Dockhorn, Franck Dernoncourt, and Lynn Gong. I also thank my Adobe office mates and friends Anthony, Chan-

Young, Darian, Dung, Gabbi, Jessica, Nham, Sean, and Steve for the great time. Eating ribs at the beach, taking photos of the redwood trees while hiking, and building LaCroix statues are memories I will never forget. I could not have asked for a better group of people to be stuck with me in a tiny conference room.

Next, I would like to thank all of the permanent and temporary members of the Harvard NLP group, Alex, Allen, Angela, Falcon, Justin, Kelly, Luke, Mirac, Rachit, Sam, Yonatan, Yoon, Yuntian, and Zach. Our reading groups were more educational than any class, and the fun I had at conferences was strongly correlated with the number of Harvard NLP members who also attended. Similar thanks goes out to all the members of the HCI group, Alex, Anna, Bernd, Elena, Joseph, Ken, Khalid, Maia, Pao, Ofra, Tianyi, Yasha, Zana, and Zilin. Thank you for bringing the human component to this work, both through the research and the outings and the free food.

Thanks to all the officemates in MD240: Bernd, for the procrastination, the post-work beers, the post-beer gym, the pink fluffy unicorns, the crypto trackers, and the sound-playing motion detectors that somehow annoyed everyone but us. Ofra, for the trips to Burdick's and the movie nights. Pao, for showing me the pleasure of sleeping at all times of day. Ken, for being the in-house tech and machine learning support who knows the answer to everything. Anna, for all the discussions that greatly inspired this dissertation. Mirac, for all the Ping Pong breaks. Also to Alex, who promptly left the office for a window-seat next door, but who still joined our procrastination sessions.

Starting graduate school, I would have never expected the mountain of bureaucratic work and the number of events that needed organizing. I would not have been able to manage those without the wonderful administrative staff, Anana, David, Darryl, Jenny, Jess, Joanne, John, Meg, and Mike. Similarly, I want to thank our custodial staff who sometimes were the only people I talked to all day when I was the only person in the office.

I also had support from my amazing roommates at 28 Marney who made sure I always came home to a fun place - Alex, Anastasiya, Angelina, Chloe, Hendrik, Lukas, Max, Patrick, Sam, Shan, and Sneha. I enjoyed our Sunday brunches, our game nights, our scary half-Christmas parties, our full-Christmas parties, and I hope that we can continue these traditions whenever we find ourselves in the same place. Thanks also to all the other friends who made this time worthwhile. Brian, for all the unexpected food breaks and for bringing his relentless happiness to every social event. John, for all the visits and fun trips to New York City. Leo, for the hikes and for driving up from Providence all the time. Michael, for the grad council events that we organized together and letting me dog-sit. Omer, for all the board and tabletop games. Wilson, for the mutual motivation to go to the gym and playing on rooftops. Seon, for the late-night beers in the dorm and hot pot. Yoon, for being the most cynical person I have ever met. Vishnu, for Thai food, Vikings, Human: Fall Flat, and generally being the center of social activities. Also thanks to all members of the Varanasi Lab at MIT for letting me join the group activities and hanging out with me. In addition, I would also like to thank all my friends in Germany who never stopped texting me, even though I don't visit often enough - Fabi, Fabi, Kjell, Lena, Luca, Luca, Nico, Nils, and Nils. I am looking forward to every visit home so we can catch up in person.

I would further like to thank my family for their support that goes back much further than graduate school. I am grateful that they drove me to all my extracurricular chess and math events. I am thankful

for supporting my decision to move (too) far away. I am extremely happy that we still get to video call and visit each other regularly. Finally, I would like to my best friend and wife Sam who has been there for me through the entirety of grad school. She commiserated when my papers got rejected, and celebrated with they got accepted. She visited me when I decided that I needed to do internships multiple hours by plane away from her, and she helped me out when the research required most of my time. But most importantly, she puts up with all my stupid puns.

# 1

## Introduction

Natural language generation (NLG) is the process of producing text from information. Typical NLG problems include the summarization of long documents and the translation from one language to another. Designing approaches for the generation of language with computers is challenging since it requires approaches that plan the content and the structure of the text and actualize it in natural, fluent, and error-free language. An autonomous system that can generate language in arbitrary contexts and across many different tasks needs to exhibit advanced reasoning skills and common-sense

knowledge. For this reason, autonomous agents that are able to generate human-like language have been regarded as the pinnacle of artificial intelligence ([Turing, 1950](#)).

A more attainable path toward autonomous NLG systems is to develop agents that aim to solve a specific generation problem, for example, a system that summarizes documents but has no ability to translate. Most current approaches to specific NLG problems are data-driven. They aim to solve the problem by learning from annotated data. Traditional data-driven approaches use a composition of independent functions to perform linguistically motivated tasks. These tasks range from the planning of the content of a text to the actualization of the plan in natural language. More recently, advances in machine learning have given rise to deep learning-based approaches. Deep learning, a family of machine learning methods based on artificial neural networks, aims to approximate the solution to a problem with a single powerful model. These complex models no longer break down a problem into stages but instead learn the entire process at once. This shift toward deep learning approaches has lead to significant performance improvements on many artificial intelligence tasks, including NLG.

As the performance of models that perform NLG tasks improves, one can imagine assistive interfaces in which a model suggests texts to writers. Consider the example of a journalist Anna who wants to write an abstract for a long news article. She could request a summarization model to write it for her, and the automation could decrease the time that she spends on writing abstracts and thus free up time to spend on research for other articles. However, journalists have a moral obligation to report accurately and objectively. If autonomous agents appropriate a part of their job, it is the journalists' duty to provide oversight over the generated texts ([Dörr and Hollnbuchner, 2017](#)). Anna, therefore, needs to trust that the generated text is accurate and follows the same structure they had in mind.

Unfortunately, deep learning models have been shown to be biased ([Caliskan et al., 2017](#)) and make potentially harmful mistakes ([Hern, 2017](#)). They are thus not trustworthy and humans who interact with automatically generated text should not trust these models.

In their learning process, deep learning models acquire undesired properties of the training data,

for example, discriminatory biases. This problem is amplified because current deep learning models are not able to explain their decisions. They only consume an input and produce a textual output that describes, translates, or compresses it. To address this problem, we need to introduce capabilities into the models that enable humans to collaborate with them in generating text. The interactions with a model need to be understandable by and intuitive to humans and help them to control the model reasoning process. To enable the collaboration with a model, we argue that its reasoning process needs to be (1) **interpretable** and (2) **controllable** by its users so that the model can act as an intelligent partner to the human.

Since current deep learning models combine all the NLG planning steps into a single function, their decisions are not always intuitive or understandable to a human reader. For a user of an assistive interface to trust a model, it is necessary that they can understand and **interpret** an explanation of how it arrives at a prediction, especially when the model makes a mistake. Only with an understandable reasoning process can users detect the inductive biases and know the limitations of the model.

Moreover, interpretability serves as a mediator for agency. As pointed out above, automation without human oversight can lead to a sharp increase in the efficiency of a task. This increase comes at the cost that humans have to give up their control over the automated process. According to Lai and Tan (2019), to gain the benefits of automation while still retaining their agency, humans have to be able to fully comprehend the model-reasoning process. Explanations can mediate between the efficiency benefits and agency retention. However, an understandable reasoning process does not retain the full agency over the automated process and is thus not sufficient for real-world tasks. Journalists, for example, are expert writers and have a particular text layout in mind that they aim to follow during the writing process. A model that automates the writing process might not follow the same plan when it generates text. That means that even if the journalist could understand the model's plan, there is no mechanism to intervene and change the reasoning process to match that of the journalist. To incorporate the feedback that the journalist has and to retain that journalists agency over the writing

process, the model needs to be **controllable**. A similar idea exists in human-robot interaction, where a robot’s planning process should be fine-tuned through a dialogue with its human controller (Fong et al., 2003). However, since deep learning models do not take discrete decisions along a reasoning chain, it is not possible to control them in ways that the more traditional models are. This leads to a disjoint nature of the human and machine writer in intelligent systems, where humans have to discard or heavily post-edit machine-suggested text if it does not match their desired text layout or phrasing. If models were controllable instead, the human writer could direct the model to follow their plan and could thus be efficiently assisted by the model.

Again, consider the journalist Anna who aims to write a summary of her article. She is familiar with the article and has an idea of what content is relevant for its summary. If she asks a model developed with current approaches to suggest the first version of a summary, this model might select a part of the article that Anna believes is unimportant, leading to a summary that she does not like. Imagine instead that she had a model that was both interpretable and controllable. Instead of a single suggestion, the model presents a first version along with an understandable visual indication of the content it evaluated as most relevant from the article. Anna can now judge for herself whether the model considered the wrong content and can provide feedback to the model, specifying the content she thinks is more important. The model, in turn, updates its reasoning process and suggests an alternative summary. This new summary matches closely with Anna’s imagined summary, which means that she only has to perform minor edits.

This thesis contributes to the interpretability and controllability of deep learning models, which can be situated in the world of collaborative agents. Researchers working on AI, HCI, and visualization have long advocated designing models as collaborative team members instead of unsupervised agents. To that end, Grosz (1996) argued that the development of “intelligent, problem-solving partners is an important goal in the science of AI”. Instead of emulating human behavior to replace people altogether, the collaborative approach can leverage the strengths of both models and humans who

work together to achieve shared goals. Terveen (1995) calls this approach the “human complementary approach” which “improves human-computer interaction by making the computer a more intelligent partner”. Collaboration centers the interaction design for the interface around the abilities of the human user instead of the model abilities. This approach is also one of the core principles of visual analytics (Keim et al., 2008). Collaborative interfaces that follow these principles could empower humans to gain the efficiency benefits of deep learning-based models while retaining control and agency over the automated process.

This dissertation argues that to enable collaboration with machine learning models, we need to consider the interaction design as an integral part of the machine learning model development. Although collaboration and teamwork are central behavioral characteristics of humans, current deep learning-based interfaces are missing these capabilities. To create intelligent autonomous partners, we advocate for the application of the design principles of mixed-initiative interfaces, which call for “mechanisms for efficient agent-user collaboration to refine results” (Horvitz, 1999). Crouser and Chang (2012) argue for a human-computer collaboration framework for joint problem-solving in which interfaces provide opportunities for actions of both human and machine agents, and that both must be able to perceive and access these actions. They define these actions, also called affordances (Gibson, 1977), as “action possibilities that are readily perceivable by a human operator”. Within the context of deep learning for language generation, the affordances are defined as possible interactions that humans can have with models. The collaboration-capabilities of a machine learning model are limited to actions that the model developer designs. This dissertation argues that we need to develop new ways to interact with models, because at present, models are neither controllable nor interpretable.

These newly designed actions could enable users to iteratively refine a proposed solution through discourse and interaction with an autonomous agent, which may lead to better results than either could achieve alone. Applying this design approach to the example from above, Anna could treat the model-generated abstract as a suggestion instead of a starting point for post-editing. She can control

the focus of the model toward essential aspects of the text and away from less salient content. The improved interaction looks more like a dialogue, going back and forth between suggestions by Anna and the model. The thesis of this dissertation is thus:

**Human-machine collaboration is necessary to effectively generate natural language.**  
**Intelligent interfaces for NLG must thus be designed to enable collaboration between**  
**humans and autonomous agents. To enable collaboration, the machine learning mod-**  
**els that power these interfaces must be inherently interpretable and controllable.**

## 1.1 THESIS OVERVIEW

This dissertation demonstrates that collaborative interfaces for NLG are necessary, possible, and effective. It first discusses what it means for an NLG model to be interpretable and how to achieve interpretability. It next introduces a mechanism that allows models to be controllable. Finally, it shows how controllable and interpretable models can be used in collaborative interfaces.

We formally introduce natural language generation problems in Chapter 2 and describe conventional approaches to these problems with a focus on deep learning.

**INTERACTION EMPOWERS HUMANS TO UNDERSTAND MODELS** In Chapters 3-6, we argue for interaction as a mechanism to make machine learning models more interpretable. Interpretability is an active area of research in machine learning. An interpretable model is defined as one that a person can understand such that they can explain predictions (Lipton, 2018). Developing an understanding of a machine learning model's behavior can be especially challenging in natural language processing (NLP) because NLP models make many sequential decisions. A goal of interpretability methods is for explanations to be presented in a way that meshes with the mental model of a person. An explanation is intuitive to the extent that it is similar to explanation a person would have given (Johnson-Laird, 1983). If the model explains its prediction in a way people do not understand, the benefit of the explanation diminishes (Narayanan et al., 2018). While this vision might seem to require explicit modeling of human reasoning processes, we accomplish the main goal through a simpler challenge. We argue that being able to interact with a model allows users to gain an understanding and develop a mental model of the machine learning model behavior instead. This understanding helps users comprehend how a model operates, explain its predictions, and recognize its limitations.

As a first step toward the goal of developing this mental model of model behavior, we need to understand the users of interpretability tools and their needs better. Therefore, in Chapter 3, we analyze

typical user types and present a distinction between model architects, trainers, and end users, who have varying knowledge about the machine learning methods and the problem domain. We further distinguish interpretability tools based on their goals. While a common goal is to understand better what a model has learned in general (model understanding), other methods target the model predictions instead (decision understanding). While these goals are not mutually exclusive, the tasks associated with each category widely differ. It is thus beneficial to focus on one of the two in a single tool. Finally, we categorize interpretability tools based on how tightly coupled the model and the interface are. For example, a model architect who aims to monitor the training process of a model does not require a tightly coupled interface. However, a model trainer who wants to understand why a model made a particular mistake requires a tighter coupled interface that enables counterfactual analyses by constraining the model predictions and changing the inputs.

We demonstrate in Chapters 4-6 that a tighter coupling leads to more powerful interactions in the interface. In three case studies of NLP, we demonstrate the need for interactive, end user-facing, interfaces that help people understand and debug the reasoning process of neural networks. In the first study, in Chapter 4, clinicians are presented with non-interactive explanations from a classification model. We show that these explanations can effectively mesh with their mental model. In the second case, in Chapter 5, we present LSTMVis, an interactive tool to identify patterns that recurrent neural networks learn. Since neural networks use high-dimensional vectors to represent their internal state, they are not inherently understandable. We show that by relating these states to textual examples that lead to a similar state, we can test whether these models learn linguistic patterns. We show how the LSTMVis interface can help users develop an intuition for what a model has learned and test hypotheses about a model’s behavior. In the third study in Chapter 6, we present Seq2Seq-Vis, an interactive debugger for neural sequence models that utilizes the understanding of model trainers to pinpoint sources of errors within a model. It is the first example that shows how inherently understandable parts of a model; in this case, the *attention*, can be used within an interface to enable effective

collaboration.

These three cases show that interaction can be used to understand a model and that an understandable model-internal reasoning process widens the possible design space for interfaces. They demonstrate that interpretability is an attainable goal for the safe and ethical application of all deep learning systems. We further observe that the described cases are restricted to the interactions that the model design allows. For example, the predictions of an end-to-end model that computes a single distribution from an input can only be understood in terms of this distribution. Since all other parts of most current deep learning models are expressed in terms of high-dimensional vectors, they do not relate to anything understandable by humans who thus cannot interact with them. Therefore, the more static and non-interpretable the model design is, the more challenging it is to enable interactions with it. In contrast, if the same model had to compute predictions of some other structured and understandable information as part of the prediction process, people could not only understand this information, but also interact with it, for example by setting or constraining its value. We thus need to change the model designs and develop inherently interactive and understandable models.

**CHAPTER 7: DISCRETE LATENT VARIABLES CAN MAKE MODELS CONTROLLABLE** A significant disadvantage of most current NLG approaches is that they are not controllable. That means that a user of an NLG tool who may want to change the length, sentiment, topic, or content of the generated text has to either reject a bad output entirely or heavily post-edit it. We argue that models can be made inherently controllable through latent variables that represent discrete decisions that the model has to make. These latent variables are tied to desired interactions that end users may want to have with the model-internal reasoning process. By exposing these discrete decisions within a user-interface, we can directly control the model inference.

Take, for example, a classification model that decides whether the language in a document is abusive. The desired interaction may be to select the words that most support the final decision. By

designing a binary latent variable for each word in the document, we are able to force the model to take this decision. Moreover, by exposing this decision within an interface, a human end user could interact with the model decision and investigate *what* would have happened *if* the model had taken a different decision.

This approach changes the order in which model and interaction design have historically been addressed. Much of the work in interaction design focuses on already established models, whereas the latent variable approach requires a model architect to think about interactions before developing the model.

We investigate the effectiveness of this approach on the problem of abstractive summarization. We extend a standard deep learning model for this problem with a mechanism that models the content selection, i.e., what words in a source document are the most relevant for its summaries. We demonstrate that modeling this decision leads to empirical improvements in the generated summaries. Moreover, we show that the content selector is much more data-efficient than a text-generating summarization model. As a result, the content selector can be trained on out-of-domain data to assist in domain transfer tasks.

**CHAPTER 8: COLLABORATIVE SEMANTIC INFERENCE** We next describe a framework for the designing of models as an intelligent partners for the human. We define the concept of collaborative semantic inference (CSI) and its place in the design space of integrating deep neural models with interactive interfaces. CSI applications deploy a dialogue process, alternating between model predictions presented in a visual form and user feedback on internal model reasoning. This process requires exposing the model’s internal process in a way that meshes with the user’s mental model of the problem and then empowering the user to influence this process.

We apply the design process to the use case of the document summarization system presented in Chapter 7 and model the content selection of the summarizer as a latent variable. We describe how

CSI can be incorporated into the model by developing a visually interactive interface that builds on the interpretability lessons identified in Chapters 3-6. By exposing the content selection decision as a controllable action, we enable a dialogue in which an end user decides which content is important, and the model generates a corresponding sentence for the summary.

In CSI, we can additionally perform inference over the latent variable decisions conditioned on the generated output. In the summarization interface, this means that the end user can write their planned summary and can see how well it covers the source document based on the latent variable decisions of the model.

## I.2 CONTRIBUTIONS

The main contributions of this dissertation are as follows:

- We develop a conceptual framework to characterize research in interpretability. We develop a categorization based on the capabilities and expertise that the users of a tool exhibit. We further classify interpretability tools based on the extent of possible model interactions and the tightness of coupling between model and interface. (Chapters 3,8)
- We demonstrate with three case studies that enabling user-interactions through tight integration of the model can empower users to develop their mental models of model behavior. (Chapter 4-6)
- We show how to generate explanations of predictions for a model that identifies patients who have specific medical conditions from textual descriptions. Clinicians rated the quality of these explanations similarly or better than comparable explanations generated for non-neural approaches. (Chapter 4)
- We introduce LSTMVis, an interactive tool to develop and investigate hypotheses about what

neural sequence models of text have learned about language. Throughout multiple use cases, we show that the hidden states within the sequence models can represent nesting depths, noun phrases, and the most common chord progressions. (Chapter 5)

- We develop `SEQ2SEQ-VIS`, a debugger for neural sequence-to-sequence models that assists in identifying what part of a model inference process has lead to a specific error. For the use case of English-German translation, we identify common sources of errors and discuss approaches to correct them. (Chapter 6)
- We demonstrate the improved performance of a document summarization system that reasons about which content is relevant for a summary. This model uses discrete latent variables as a means to make natural language generation controllable. (Chapter 7)
- We develop the concept of collaborative semantic inference as a framework to develop collaborative intelligent interfaces. We show how the framework can be applied to develop a collaborative summarization interface. (Chapter 8)

*The Analytical Engine has no pretensions to originate anything. It can do whatever we know how to order it to perform.*

Ada Lovelace

# 2

## Natural Language Generation

NATURAL LANGUAGE GENERATION (NLG) is the process of generating fluent, coherent, and accurate language that describes data. These data can come in many different forms, ranging from tabular data to long documents and images. In order to produce this language, a natural language generation algorithm needs to identify *what* to say and then decide *how* to say it. In addition to the input, the generation process depends on the task and the intended audience, a process that answers *why* a sys-

tem should phrase something in a certain way (Hovy, 1987). For example, a system that is deployed in a hospital and aims to summarize information from the medical record should use technical details when the audience are doctors and nurses, but use simple and understandable language when communicating with patients (Mahamood and Reiter, 2011).

The traditional approach to deal with this challenging process is to structure and break up an NLG problem into a series of smaller subproblems. A common structure is composed of the following three steps as defined by Reiter (1994)<sup>1</sup>:

1. **Content Determination and Text Planning** Decide what information should be communicated and how to structure it rhetorically.
2. **Sentence Planning** Decide how to break the information into individual sentences.
3. **Realization** Actualize the abstract representation in natural language.

The rise of deep learning approaches to NLG problems lead to the ubiquitous use of so-called end-to-end methods where the entire task is learned with a complex neural network (Dušek et al., 2020). With some exceptions (e.g., Elder et al., 2019, Fan et al., 2019), these approaches do not (intentionally) follow the same structure, disconnecting them from the human-like traditional approaches. As noted in the previous chapter, this disconnect leads to many challenges, which we aim to address in this dissertation.

Before addressing these challenges, we provide an introduction to common problem formulations in NLG and an overview of deep learning methods to approach them in this section. In Section 2.1, we introduce the notation we follow throughout this dissertation. In Section 2.2, we formalize the most common language generation tasks. In Section 2.3, we review deep learning approaches for NLG.

Finally, in Section 2.4, we briefly discuss inference methods that are used for these models.

---

<sup>1</sup>In later work, Reiter and Dale (1997, 2000) break the three steps down into more fine-grained subcategories. For a more up-to-date overview of these categories, we refer to the description by Gatt and Krahmer (2018).

## 2.1 NOTATION

In the supervised text-to-text problem, let  $(\mathbf{x}^{(0)}, \mathbf{y}^{(0)}), \dots (\mathbf{x}^{(N)}, \mathbf{y}^{(N)}) \in (\mathcal{X}, \mathcal{Y})$  be a set of  $N$  aligned source and target sequence pairs, with  $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$  denoting the  $i$ th element in  $(\mathcal{X}, \mathcal{Y})$ . Throughout this work, we use indices to indicate positional information within a sequence. For a problem that uses text as input to produce text as output, let  $\mathbf{x} = x_1, \dots, x_S$  be the sequence of  $S$  tokens in the source, and  $\mathbf{y} = y_1, \dots, y_T$  the target sequence of length  $T$ . Whenever a feature is represented as a vector, we use bold face, for example when we have a sequence of vectorized features in an input  $\mathbf{x}$ , we refer to the feature representation of its tokens as  $\mathbf{x}_1, \dots, \mathbf{x}_S$ . We use lower case letters to refer to vectors and upper case letters to refer to matrices, for example bias vectors  $\mathbf{b}$  and weight matrices  $\mathbf{W}$ . We will further denote the list of integers between  $i$  and  $j$  as  $i:j$ . As a shorthand, we use  $< j$  to refer to the list of integers from 1 to  $j-1$ .

## 2.2 NATURAL LANGUAGE GENERATION TASKS

Probabilistic models for conditional generation tasks aim to learn a distribution parametrized by  $\theta$  to maximize the conditional probability of  $p_\theta(\mathbf{y}|\mathbf{x})$ . We typically assume that the target is generated from left to right with an autoregressive model, such that

$$p_\theta(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^T p_\theta(y_t | \mathbf{y}_{<t}, \mathbf{x}) = \sum_{t=1}^T \log p_\theta(y_t | \mathbf{y}_{<t}, \mathbf{x}),$$

A single prediction step involves the computation of  $p_\theta(y_t | \mathbf{y}_{<t}, \mathbf{x})$ , which can take the form of various models that we introduce in Section 2.3.

Assuming that we have access to annotated training data, the training process aims to minimize the negative log-likelihood of the observed training data,

$$-\sum_{i=1}^N \sum_{t=1}^{|\mathbf{y}^{(i)}|} \log p_\theta(y_t^{(i)} | \mathbf{y}_{<t}^{(i)}, \mathbf{x}^{(i)}).$$

With a fully differentiable model architecture, this likelihood can be optimized using derivative-based methods such as stochastic gradient descent and its variants. Since optimization is not focus of this dissertation, we refer to [Ruder \(2016\)](#) for an overview of these approaches.

### 2.2.1 LANGUAGE MODELING

The color of the dog is [brown/round/was/...]

**Figure 2.1:** In language modeling, the task is to predict the next word under a given context from a vocabulary of all the words in the language. In this example, a plausible next word is *brown*.

Language modeling is the problem of predicting the word that follows a context of preceding words, as shown in Figure 2.1. Language modeling does not technically qualify as an NLG problem, since language models (LM) do not produce text from data, but only from samples. It does, however, embody the core challenge of language generation of generating words.

In language modeling, at a time step  $t+1$ , the prefix of words  $x_1, \dots, x_t$  is taken as input and the goal is to model the distribution over the next word  $p(x_{t+1}|x_1, \dots, x_t)$ . The specific task was developed by [Shannon \(1948\)](#) who used this task to estimate a bound on the entropy of English. Historically, LMs were of crucial importance for many tasks in NLP, for example by assigning likelihoods of word-sequences in machine translation ([Brown et al., 1988](#)) and speech recognition ([Jelinek, 1997](#)).

### 2.2.2 TRANSLATION

With a history almost as longstanding as language modeling ([Dostert, 1955](#)), machine translation is the task of automatically translating from one natural language to another. In machine translation,  $\mathbf{x}$

German: Die Farbe des Hundes ist braun.  
English: The color of the dog is [brown/round/was/...]

**Figure 2.2:** In machine translation, the task is to predict the next word under a given context from a vocabulary of all the words in the language. This decision is further conditioned on the same sentence expressed in another language.

denotes a sequence of tokens in one language and  $\mathbf{y}$  a sequence of tokens in another language. Since not all languages share the same syntax, a model has to successfully learn an alignment between the two languages. While traditionally, statistical approaches learned separate language and alignment models, deep-learning approaches combine these steps into a single end-to-end model. The alignment represents the input that is most relevant for the current translation step and can be formalized as a distribution over the input tokens. We denote this distribution which is commonly called the *attention* as  $p_{attn}(\mathbf{a}_t | \mathbf{x}, \mathbf{y}_{<t})$  for a decoding step  $t$ . The prediction which word should follow uses this model-intrinsic attention distribution to compute the distribution over the next word  $p(y_t | \mathbf{y}_{<t}, \mathbf{x})$ . In the example presented in Figure 2.2, an alignment model could predict that the next word should translate the German word *braun* by assigning a high attention weight. Through this information, the model determines that the next word should be its English translation, *brown*.

### 2.2.3 SUMMARIZATION

In summarization, we consider a set of documents where  $\mathbf{x}$  corresponds to tokens  $x_1, \dots, x_S$  in a source document  $\mathbf{y}$  to a summary  $y_1, \dots, y_T$  with  $T \ll S$ . In contrast to translation, both source and summary share the same language. The goal of a summary is to convey the same information as the long source document. These summaries aim to describe the so-called macrostructure, or the coherence, of the discourse in the source document (Van Dijk, 1977, Seidlhofer, 1995).

Summarization is typically divided into extractive and abstractive approaches. In extractive summarization, the model is constrained to only use sentences or phrases from the input document. In abstractive summarization, a model is allowed to paraphrase and reformulate the words from the in-

Long Input:	There exist a brown canine which has been observed to have a brown color. It runs around the yard, in search for its lost ball.
Sentence Extractive:	There exist a brown canine which has been observed to have a brown color.
Phrase Extractive:	A brown canine which has a brown color runs around the yard.
Abstractive:	A brown dog searches for its ball.

**Figure 2.3: Summarization** describes the task of compressing the information in a long document in short and fluent language. Extractive summaries are composed of a subset of the sentences or phrases from the input, whereas abstractive summaries can use paraphrase and reformulate the input.

put. Similar to machine translation, abstractive summaries are generated one word at a time. At every time-step, a model is aware of the previously generated words and can utilize an alignment to the input.

While the extractive summarization task is much easier, abstractive summaries can be much more concise. In the example presented in Figure 2.3, the abstractive summary can paraphrase *canine* as *dog* and compress the relative clause into a single adjective *brown*. For that reason, most neural approaches try to combine the advantages of both approaches by incorporating mechanisms for abstractive and extractive summarization behavior. In end-to-end models, the most common approach is to use a so-called *copy attention*. The copy attention follows the same structure as the standard attention by computing an alignment between the words in the source document and the current decoding step. The attention weights, in conjunction with a predicted probability to copy a word, are used to determine whether and which input word should be copied into the summary draft.

#### 2.2.4 DATA-TO-TEXT

Input: color(brown), animal(dog).  
 Output: The color of the dog is brown.

**Figure 2.4: Data-to-text** problems are those without textual input. In this example, the input are key-value attribute pairs. The goal of this task is to describe these attributes.

The problems we introduced up to this point are text transduction tasks where one sequence of

words is used as input to generate another. Data-to-text problems are those where the input is not fluent text (Gatt and Krahmer, 2018). Within the class of data-to-text problems, we focus on the problem of generating text from structured key-value attribute pairs as shown in Figure 2.4. This task is related to the traditional NLG pipeline, as the output of the intermediate planning steps are often *abstract meaning representations* (MR) that take this form.

The problem treated as the same transduction task introduced above by mapping the structured data into a sequential form. In order to apply the same end-to-end models used for summarization, a list of attributes in an MR has to be linearized into a sequence of tokens (Konstas et al., 2017, Ferreira et al., 2017). For example, an approach could map an attribute *color(brown)* into the sequence `_start_color_ brown _end_color_` (Gehrmann et al., 2018a).

### 2.2.5 OTHER NLG TASKS

There exists a myriad of other NLG tasks that are not included in the work discussed in this dissertation, each of which has their own challenges and uses. However, deep learning is currently the standard approach to all of these problems, at least in research. They thus suffer from many of the same issues we discuss throughout this dissertation.

A notable task is that of dialogue generation, or conversational AI. Dialogue was the original task posed by Turing (1950) to assess whether an artificial agents behavior is indistinguishable from that of a human. The problems we introduced so far all map from one input to one output. Models for conversational AI need to solve the same type of text-transduction problem, but they also need to be aware of everything that has been said before and of the overall dialogue structure, and they need to be able to respond to arbitrary prompts. The task is thus much more challenging than summarization or translation.

There also exist NLG tasks for input of completely different modalities, for example image or video captioning. Models that address these tasks need a structure that is able to encode these inputs, but

otherwise looks strikingly similar to the end-to-end models we discuss. In addition, there also exist tasks with multiple input modalities at the same time, for example visual question answering, where a model should answer a question about an image.

Despite their differences, the deep learning approaches that are utilized to these problems share many similarities and are composed of the same building blocks. We describe these building blocks in the following section.

### 2.3 DEEP LEARNING FOR NATURAL LANGUAGE PROCESSING

Most recent advances in natural language generation apply deep learning, a family of approaches that is composed of neural network-based methods. In this section, we review the deep learning methods that are applied to language generation problem and which we will use throughout the rest of this dissertation.

#### 2.3.1 WORD EMBEDDINGS

Features in NLP systems have traditionally been task-specific and hand-designed and were represented as one-hot vectors  $\{0, 1\}^D$ , where  $D$  is the number of features. Let  $\delta(x_t) \in \mathbb{R}^{D \times 1}$  be the representation of the  $t$ -th token within a sentence  $\mathbf{x}$ . Consider a vocabulary with the six words

$$\{\text{a, cat, dog, jumps, runs, the}\}$$

and the phrase *the dog jumps*. The sentence could then be represented as a sequence of vectors:

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

However, if you consider the semantically and syntactically very similar phrase *a cat runs*, you notice that the two phrases have completely disjoint representations<sup>2</sup>.

The intuition behind word embeddings is that representing words or features as a dense real-valued feature vector can capture these similarities, especially when the embedding can be trained jointly with the rest of a model (Bengio et al., 2003, Collobert and Weston, 2008, Collobert et al., 2011). Words that occur in similar contexts learn similar embeddings. Since misspellings, synonyms and abbreviations of a word occur in similar contexts, a database of synonyms and common misspellings is not required (Carrell et al., 2014). To map from  $\delta(x_t)$ , consider an *embedding matrix*  $\mathbf{W}_{emb} \in \mathbb{R}^{D_{emb} \times D}$ . By multiplying  $\mathbf{W}_{emb} \times \delta(x_t)$ , we are picking the  $x_t$ -th column of  $\mathbf{W}_{emb}$ , as illustrated in the following example:

$$\begin{bmatrix} | & | & | \\ w_1 & w_2 & w_3 \\ | & | & | \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} | \\ w_3 \\ | \end{bmatrix}$$

Another advantage of distributed representations of words is that they can be used to leverage large corpora of unlabeled text. Since language modeling does not require any annotated data, models and their embeddings can be trained on much larger corpora than other tasks. Given the intuition that an embedding reflects the meaning of a word, Collobert and Weston (2008) found that transferring  $\mathbf{W}_{emb}$  from a language model to other tasks significantly improved the performance on those tasks.

---

<sup>2</sup>The example is inspired by Bengio et al. (2003), who pioneered the use of word embeddings in language models.

In later work, Mikolov et al. (2013) and Pennington et al. (2014) introduced word2vec and GloVe respectively, which are both methods to learn meaningful word embeddings. The resulting embeddings have shown to improve performance across many NLP tasks, especially for tasks with limited training data (Erhan et al., 2010, Luan et al., 2015, Wu et al., 2015).

Further expanding the idea of unsupervised representation learning, McCann et al. (2017) introduced the idea of *contextualized word embeddings*. The idea of these embeddings is that the characteristics of a word can vary depending on its linguistic context<sup>3</sup>. A contextualized word embeddings is a function of a wide context to account for the variation, while still benefiting from unsupervised training. While CoVe by McCann et al. (2017) used the representations learned by a machine translation system, Peters et al. (2018) introduce ELMo, which derives the representation from a bidirectional language model. More recently, Devlin et al. (2019) introduced BERT, which explicitly learns entailment in addition to language modeling.

### 2.3.2 MULTI-LAYER PERCEPTRONS

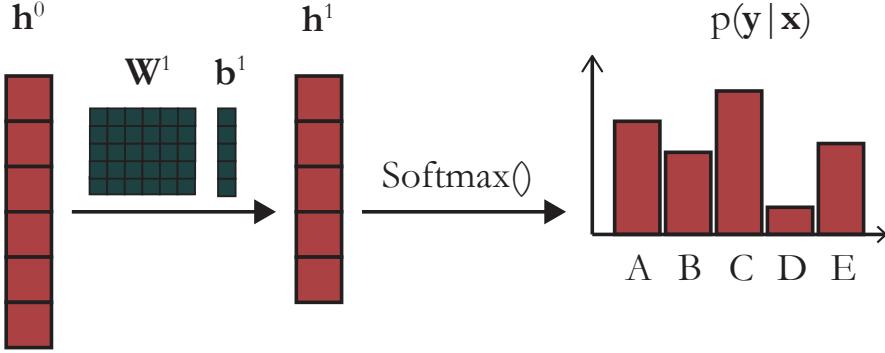
Recall a logistic regression which is a linear classification model. Suppose we have a  $D$ -dimensional input  $x$  and  $C$  potential classes in  $y$ <sup>4</sup>. A logistic regression defines a scoring function  $\mathbf{LR}(x) = \mathbf{W}x = \sum_{i=1}^D \mathbf{W}_i x_i$ , where  $\mathbf{W} \in \mathbb{R}^{C \times D}$  is a learnable parameter.  $\mathbf{LR}(x)$  yields  $C$  scores, one for each class. We can then apply the softmax function to transform the scores, also called logits, into normalized probabilities

$$p(y=i|x) = \frac{\exp(\mathbf{LR}(x)_i)}{\sum_{c=1}^C \exp(\mathbf{LR}(x)_c)}.$$

---

<sup>3</sup>Consider the sentences *The children play in the leaves* and *Their father leaves for work*. Since the word *leaves* is a homonym, it can be either a verb or a noun and completely switch meaning depending on the context. A non-contextualized word embedding would assign the same embedding to both.

<sup>4</sup>It is typical that  $y$  is represented as one-hot vector  $\{0, 1\}^C$  with a 1 as indicator for the correct class.



**Figure 2.5:** An illustration of a Perceptron model. A linear transformation is applied to the representation  $\mathbf{h}^0$  to generate the new state  $\mathbf{h}^1$ . This is followed by a softmax-function to generate a distribution over possible outputs  $p(\mathbf{y}|\mathbf{x})$ .

The Perceptron (Rosenblatt, 1957, Minsky and Papert, 1969) additionally introduces a bias term  $\mathbf{b} \in \mathbb{R}^C$  that is added to  $\mathbf{W}x$ <sup>5</sup>.

The Multi-Layer Perceptron (MLP), also called the vanilla neural network, introduces non-linearity in the form of hidden layers. MLPs are chains of non-linear transformations that aim to learn highly complex functions. Consider the following one-layer MLP:

$$\text{MLP}_1(x) = \mathbf{W}^2 \underbrace{g(\mathbf{W}^1 x + \mathbf{b}^1)}_{\text{One ‘layer’}} + \mathbf{b}^2.$$

Here,  $\mathbf{W}^1 \in \mathbb{R}^{D_{hid} \times D}$ ,  $\mathbf{b}^1 \in \mathbb{R}^{D_{hid}}$ ,  $\mathbf{W}^2 \in \mathbb{R}^{C \times D_{hid}}$ , and  $\mathbf{b}^2 \in \mathbb{R}^C$ . The one-layer MLP comprises two linear Perceptron-like transformations, intersected by a non-linear function  $g$ . The activation function  $g$  can take different forms such as the sigmoid-function, tanh, or *ReLU* (Nair and Hinton, 2010). Its result is also called the hidden state  $\mathbf{h}$ .

By recursively defining a layer as mapping from one vector  $\mathbf{h}^i$  to its successor  $\mathbf{h}^{i+1}$  such that  $\mathbf{h}^{i+1} = g(\mathbf{W}^{i+1}\mathbf{h}^i + \mathbf{b}^{i+1})$ , we observe that we can combine an arbitrary number of transformations, or layers, within the neural network. In this definition,  $\mathbf{h}^1$  represents the input to the network, for example a

---

<sup>5</sup>As Goldberg (2016) points out, a bias term can be simulated in a logistic regression via an additional dimension in  $x$  whose value is always 1.

word embedding.

### 2.3.3 CONVOLUTIONAL NEURAL NETWORKS

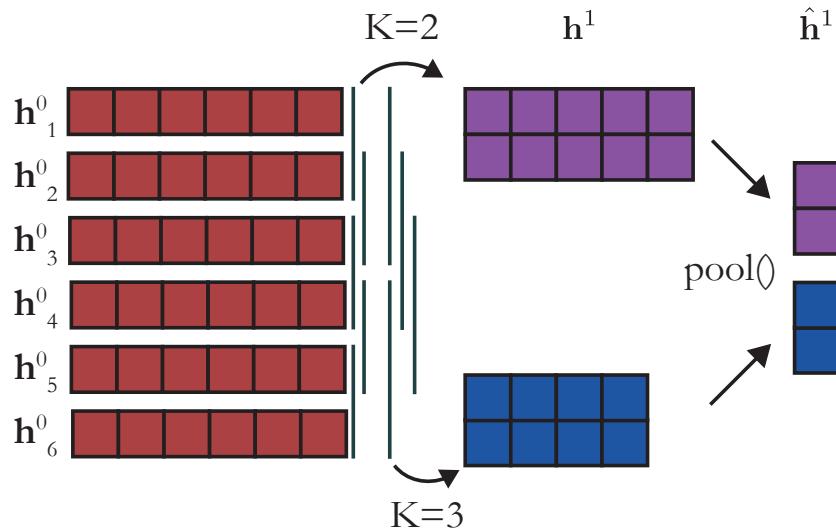
So far, we have introduced dense representations of words and a way to derive features from them via linear layers in a neural network. Convolutional Neural Networks (CNNs) represent a way to derive features from sequences of dense representations (LeCun et al., 1998). This approach is particularly helpful for classification tasks, as identified by Collobert et al. (2011) and Kim (2014). The idea behind convolutions stems from computer vision, where the goal is to learn “filters” that transform adjacent pixels into single values. Equivalently, a CNN for NLP learns which combinations of adjacent words or features are associated with a given concept.

To represent a sequence of features, consider a sequence of  $T$  features,  $\mathbf{h}_1^i, \dots, \mathbf{h}_T^i$ , where  $\mathbf{h}_t^i \in \mathbb{R}^{D_i}$ . A convolutional operation applies  $D_{i+1}$  filters of trained parameters  $\mathbf{W} \in \mathbb{R}^{D_{i+1} \times K D_i}$  to an input-window of concatenated feature representations with a width of  $K$ . Therefore, a sequence of word embeddings can be transformed as

$$\mathbf{h}_j^{i+1} = g(\mathbf{W} \underbrace{[\mathbf{h}_t^i, \dots, \mathbf{h}_{t+K-1}^i]}_{\text{the current window of inputs}} + \mathbf{b}),$$

where  $\mathbf{b}$  represents another bias term and  $g$  a non-linearity and which is applied for  $t=1, \dots, T-K+1$ . Since this operation yields  $T-K+1$  representations in  $\mathbb{R}^{D_{i+1}}$ , we may want to combine them into a single feature  $\hat{\mathbf{h}}$ . This is typically done via *pooling* operations, which can average these representations or pick the max/min values in each dimensions. In this context, max-pooling, the most commonly applied operation in NLP, is defined as

$$\hat{\mathbf{h}}_d^{i+1} = \max \underbrace{\left\{ \mathbf{h}_{1,d}^{i+1}, \dots, \mathbf{h}_{T-K+1,d}^{i+1} \right\}}_{\text{the derived feature maps}}$$



**Figure 2.6:** We illustrate a simple CNN model with two filters of size 2 and 3. These filters are applied to a sequence of input representations  $\mathbf{h}^0$  to generate the features  $\mathbf{h}^1$ . Through a pooling operation, each dimension in  $\mathbf{h}^1$  is reduced to a single value, also denoted as  $\hat{\mathbf{h}}^1$ .

for  $d=1, \dots, D_{i+1}$ . Given these functions, we can construct a simple CNN-based network to classify text, as shown in Figure 2.6. In the example, each word is represented as its embedding  $\delta(x_t)$ . The sequence of embeddings are used as  $\mathbf{h}^1$  to which the convolutional operations are applied. In the example, we apply multiple convolutional operations of different widths ( $K = 2, \dots, 5$ ) and  $D^{i+1}=1$  in parallel to capture information with different phrase-lengths. The result of the pooling operations is used within a final linear layer to capture the classification probability.

#### 2.3.4 RECURRENT NEURAL NETWORKS

While CNNs can effectively learn features from multiple consecutive inputs, they are fundamentally limited by the width of the convolutional filters. Often, the information that are relevant for a prediction are outside of the current window and thus cannot be taken into account by the model. Recurrent neural networks (RNN, Elman, 1990) are an approach to make neural networks stateful, and to enable the representation of sequences of arbitrary length.

RNNs are a class of neural networks that sequentially map representations such as word embeddings  $\mathbf{h}_1^i \dots \mathbf{h}_T^i$  to a sequence of contextual hidden feature-state representations  $\mathbf{h}_1^{i+1} \dots \mathbf{h}_T^{i+1}$ . This is achieved by learning the weights of a neural network **RNN**, which is applied recursively at each time-step  $t = 1, \dots, T$ :

$$\mathbf{h}_t^{i+1} = \text{RNN}(\mathbf{h}_t^i, \mathbf{h}_{t-1}^{i+1})$$

The **RNN** function takes input vector  $\mathbf{h}_t^i$  and a hidden state vector  $\mathbf{h}_{t-1}^{i+1}$  and gives a new hidden state vector  $\mathbf{h}_t^{i+1}$ . Similar to the related neural architectures, each hidden state vector  $\mathbf{h}_t^{i+1}$  is in  $\mathbb{R}^{D_{i+1}}$ . There exist many explicit formulation of the function. The simplest form, often called the vanilla RNN, transforms the two inputs via linear transformations, followed by a non-linearity,

$$\mathbf{h}_t^{i+1} = \tanh\left(\underbrace{\mathbf{W}^1 \mathbf{h}_t^i}_{\text{current input}} + \underbrace{\mathbf{W}^2 \mathbf{h}_{t-1}^{i+1}}_{\text{previous hidden state}} + \mathbf{b}\right),$$

with  $\mathbf{W}^1 \in \mathbb{R}^{D_{i+1} \times D_i}$ ,  $\mathbf{W}^2 \in \mathbb{R}^{D_{i+1} \times D_{i+1}}$ , and  $\mathbf{b} \in \mathbb{R}^{D_{i+1}}$ . Long Short-Term Memory recurrent neural networks (LSTM, [Hochreiter and Schmidhuber, 1997](#)), define a variant of RNN that has a modified hidden state update which can more effectively learn long-term interactions. LSTMs maintain both a cell state vector and a hidden state vector at each time step. These vectors are modulated by three gates, the input-, forget-, and output-gate, which are defined as follows<sup>6</sup>:

$$\begin{aligned}\mathbf{i}_t &= \sigma(\mathbf{W}^{i1} \mathbf{h}_t^{i-1} + \mathbf{W}^{i2} \mathbf{h}_{t-1}^i + \mathbf{b}^i) \\ \mathbf{f}_t &= \sigma(\mathbf{W}^{f1} \mathbf{h}_t^{i-1} + \mathbf{W}^{f2} \mathbf{h}_{t-1}^i + \mathbf{b}^f) \\ \mathbf{o}_t &= \sigma(\mathbf{W}^{o1} \mathbf{h}_t^{i-1} + \mathbf{W}^{o2} \mathbf{h}_{t-1}^i + \mathbf{b}^o)\end{aligned}$$

---

<sup>6</sup>For brevity, we assume an input feature  $h^{i-1}$  instead of  $h^i$ .

All three gates are moderated by the sigmoid function to ensure that their values lie in  $[0, 1]$ . The values are then used to construct the new cell state  $\mathbf{c}_t^i$  and the new hidden state  $\mathbf{h}_t^i$ ,

$$\begin{aligned}\tilde{\mathbf{c}}_t^i &= \tanh(\mathbf{W}^{c1}\mathbf{h}_t^{i-1} + \mathbf{W}^{c2}\mathbf{h}_{t-1}^i + \mathbf{b}^c) \\ \mathbf{c}_t^i &= \mathbf{f}_t\mathbf{c}_{t-1}^i + \mathbf{i}_t\tilde{\mathbf{c}}_t^i \\ \mathbf{h}_t^i &= \tanh(\mathbf{c}_t^i\mathbf{o}_t)\end{aligned}$$

Here,  $\tilde{\mathbf{c}}_t^i$  represents the actual update of the cell state  $\mathbf{c}_t^i$ , and the input-gate represents how much of it gets written to the cell. Similarly, the forget-gate decided how much of the previous steps cell-state is “remembered”. The output-gate decides how the cell-state should be used in the new hidden state.

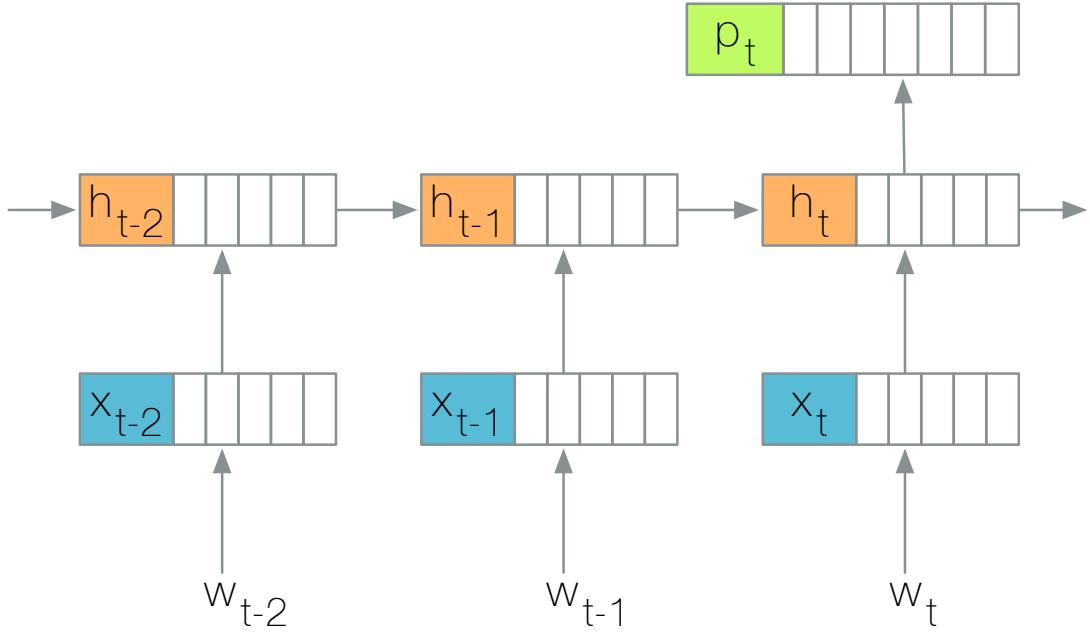
RNNs are one of the most common approach to sequence-modeling problems, such as language modeling (Mikolov et al., 2010, Zaremba et al., 2014). To utilize an RNN to produce a probability distribution in this multi-class classification setting with  $C$  classes<sup>7</sup>, we need to transform the hidden feature-state vector  $\mathbf{h}_t^i$ , which lies in  $\mathbb{R}^{D_i}$ . Formally we define this as

$$p(x_{t+1} | \underbrace{x_1, \dots, x_t}_{\text{the context}}) = \text{softmax}(\mathbf{W}\mathbf{h}_t + \mathbf{b}),$$

where  $\mathbf{W} \in \mathbb{R}^{D_i \times C}$ ,  $\mathbf{b} \in \mathbb{R}^C$  are further parameters. The full computation of an RNN language model is illustrated in Figure 2.7. A common extension to the RNN formulation is to represent a left and a right context to two different RNNs respectively and to concatenate the resulting representations. This approach is called the bidirectional RNN, or BiRNN for short. BiLSTM based approaches have been the most prevalent approach to encode many of the inputs for the described NLG tasks.

---

<sup>7</sup>In the language modeling example,  $C$  is equal to the vocabulary size, which is also often denoted as  $V$ .



**Figure 2.7:** A recurrent neural network language model being used to compute  $p(x_{t+1}|x_1, \dots, x_t)$ . At each time step, a word  $x_t$  is converted to a word vector  $\delta(x_t)$ , which is then used as initial feature vector  $h_t^1$  to update the hidden state  $h_t^2 = \text{RNN}(h_t^1, h_{t-1}^2)$ . This hidden state vector can be used for prediction. In language modeling it is used to define the probability of the next word,  $p(x_{t+1}|x_1, \dots, x_t) = \text{softmax}(\mathbf{W}h_t^2 + \mathbf{b})$ .

RNNs learn to utilize the hidden states to represent the features of the input, but due to the non-linear transformations and high-dimensional representations, they are difficult to understand. The dynamics of the hidden states in RNNs, particularly their change over time, are a central point of investigation in Chapter 5.

### 2.3.5 SEQUENCE-TO-SEQUENCE MODELS AND ATTENTION-MECHANISMS

The methods we described so far are limited to learning to generate representations of sequential inputs. However, often we care about transducing text, where the sequential output is additionally conditioned on input. Moreover, in those cases we care about modeling interactions between the input and output representations. This problem is approached with sequence-to-Sequence (seq2seq)

models (Bahdanau et al., 2015, Sutskever et al., 2014). Seq2seq models can be broken down into four stages: an *encoder* and *decoder*, the *attention*, and a *prediction*.

Recall the machine translation example from Chapter 2.2.2, which aims to translate the German sentence “Die Farbe des Hundes is braun” and the partial translation “The color of the dog is ...”. The encoder, typically some variant of the RNN or BiRNN, is used to encode the source into a sequence of hidden representations  $\mathbf{x}_1, \dots, \mathbf{x}_S$ . The decoder, which typically also assumes the form of a RNN<sup>8</sup>, is used to encode the current state of a translation  $\mathbf{y}_1, \dots, \mathbf{y}_{t-1}$ .

To arrive at the correct next word, the network next computes an attention between the input  $\mathbf{x}$  and the current decoding step  $y_t$ , which is further conditioned on the previously translated words  $\mathbf{y}_{<t}$ . In the example, the attention mechanism focuses on the German word “braun” so that the network can predict its translation “brown”.

**ATTENTION** The attention mechanism comprises a family of approaches that aim to align a representation to a sequence and to represent a measure of focus within a neural network. Within the network, the attention distribution  $p(\mathbf{a}_t | \mathbf{x}, \mathbf{y}_{<t})$  for a decoding step  $t$  represents an embedded soft distribution over all of the source tokens.

The attention first computes an alignment score for each source token according to a scoring function  $\text{score}(\mathbf{x}, \mathbf{h}_t)$  that uses the current decoder hidden state  $\mathbf{h}_t$  as input. This score is then converted into a distribution such that  $p(\mathbf{a}_t | \mathbf{x}, \mathbf{y}_{<t}) = \text{softmax}([\text{score}(\mathbf{x}, \mathbf{h}_t)_1, \dots, \text{score}(\mathbf{x}, \mathbf{h}_t)_S])$ . Given the attention distribution, we compute a single vector that represents the entire context over which the network is attending. The context vector is a sum of hidden states in the encoder  $\mathbf{x}_s$ , weighted by their attention weight  $a_{t,s}$ ,

---

<sup>8</sup>Since autoregressive decoding predicts words in a left-to-right manner, BiRNNs cannot be applied in a decoder.

$$\mathbf{c}_t = \sum_s^S a_{t,s} \mathbf{x}_s.$$

For RNN-based sequence-to-sequence models, there exist two popular variants of the scoring function. The first uses a dot-product between the hidden states of the encoder and decoder (Luong et al., 2015). The second uses a multi-layer perceptron with the hidden states as inputs (Bahdanau et al., 2015). For each encoder token, the scores are defined as

$$\begin{aligned} \text{score}_{dot}(\mathbf{x}, \mathbf{h}_t)_s &= \mathbf{x}_s \cdot \mathbf{h}_t \\ \text{score}_{MLP}(\mathbf{x}, \mathbf{h}_t)_s &= \mathbf{W}_{attn} \tanh (\mathbf{W}_s \mathbf{x}_s + \mathbf{W}_h \mathbf{h}_t + \mathbf{b}_{attn}), \end{aligned}$$

where all **Ws** and **bs** are trainable parameters.

**PREDICTION** In seq2seq models, the context vector is next used to compute an output representation  $\mathbf{h}_t^{out}$ . Let  $[\mathbf{h}_t, \mathbf{c}_t]$  denote the concatenation of the decoder hidden state and the context vector. This can be used to calculate an intermediate representation  $\mathbf{h}_t^{out} \in \mathbb{R}^{D_{hid}}$  as

$$\mathbf{h}_t^{out} = \mathbf{W}_{out} [\mathbf{h}_t, \mathbf{c}_t] + \mathbf{b}_{out}.$$

Finally, a generator calculates the conditional probability distribution over the vocabulary  $\mathcal{V}$ ,

$$p(y_t | \mathbf{y}_{<t}, \mathbf{x}) = \text{softmax}(\mathbf{W}_{gen} \mathbf{h}_t^{out} + \mathbf{b}_{gen}).$$

In those computations,  $\mathbf{W}_{out} \in \mathbb{R}^{D_{hid} \times 2 \cdot D_{hid}}$ ,  $\mathbf{b}_{out} \in \mathbb{R}^{D_{hid}}$ ,  $\mathbf{W}_{gen} \in \mathbb{R}^{|\mathcal{V}| \times D_{hid}}$ , and  $\mathbf{b}_{gen} \in \mathbb{R}^{|\mathcal{V}|}$  are all trainable parameters.

**COPY ATTENTION** In some NLG problems, parts of the inputs should be replicated in the output. For example, when the goal is to describe a restaurant based on key-value attributes, it is beneficial to give the model the ability to copy for example the restaurant name which otherwise would not be part of  $\mathcal{V}$ . In these cases, the seq2seq model can be augmented with a copy mechanism (Vinyals et al., 2015, Gu et al., 2016) to copy words from the source. Copy models extend the decoder by predicting a binary soft switch  $z_t$  that determines whether the model copies or generates. The copy distribution is a probability distribution over the source text, and the joint distribution is computed as a convex combination of the two parts of the model,

$$\begin{aligned} p(y_t | \mathbf{y}_{<t}, \mathbf{x}) = \\ p(z_t = 1 | \mathbf{y}_{<t}, \mathbf{x}) \times p(y_t | z_t = 1, \mathbf{y}_{<t}, \mathbf{x}) \\ + p(z_t = 0 | \mathbf{y}_{<t}, \mathbf{x}) \times p(y_t | z_t = 0, \mathbf{y}_{<t}, \mathbf{x}) \end{aligned}$$

where the two parts represent copy and generation distribution respectively.

The computation of  $p(z_t | \mathbf{y}_{<t}, \mathbf{x})$  uses the same output representation that is used as input to the generator,

$$p(z_t | \mathbf{y}_{<t}, \mathbf{x}) = \sigma(\mathbf{W}_{\text{copy}} \mathbf{h}_t^{\text{out}} + b_{\text{copy}}),$$

with  $\mathbf{W}_{\text{copy}} \in \mathbb{R}^{1 \times D_{\text{hid}}}$  and  $b_{\text{copy}} \in \mathbb{R}$ , and the  $\sigma$ -function ensuring values between 0 and 1<sup>9</sup>. The other addition in the copy mechanism is the copy distribution  $p(y_t | z_t = 1, \mathbf{y}_{<t}, \mathbf{x})$ . The copy distribution is a distribution over the input which can be computed with a separate module as first introduced by Gu et al. (2016). However, most recent work reuses the attention distribution  $p(\mathbf{a}_t | \mathbf{x}, \mathbf{y}_{<t})$  which leads to the same or better results and avoids the additional computational overhead. To trans-

---

<sup>9</sup>Some papers, for example See et al. (2017), use a slightly different computation with additional inputs, but our empirical results show no difference in performance.

late the attention distribution to a distribution over a vocabulary, we add the incoming attention weights for each source token. That means, if an input has two tokens “dog” with attention weights 0.1 and 0.3 respectively,  $p(y_t = \text{dog} | z_t = 1, \mathbf{y}_{<t}, \mathbf{x}) = 0.4$ . Since an input token could be not part of  $\mathcal{V}$ , or out-of-vocabulary, this operation can lead to a high probability for previously unknown tokens with probability 0.

### 2.3.6 TRANSFORMER

To further build on the strong performance of attention-based models, [Vaswani et al. \(2017\)](#) developed the Transformer architecture which represents an alternative to recurrent neural networks. Instead of using recurrence to model the context of a token, the Transformer uses *self-attention*, an attention that is computed from a token within a sequence over the sequence itself. Recall the computation of the context vector following the scoring function by [Luong et al. \(2015\)](#), Since,  $\mathbf{h}_s$  and  $\mathbf{x}_s$  would be equal in the unmodified self-attention, the Transformer extends this formulation. Consider the following interpretation:  $\mathbf{h}_t$  is a query that looks for the most similar values in they key  $\mathbf{X}$ , where  $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_S]$ . It then uses this similarity to weigh the underlying value, which in the default case is also  $\mathbf{X}$ ,

$$\mathbf{c}_t = \text{softmax} \left( \underbrace{\mathbf{X}}_{\text{Key}} \cdot \underbrace{\mathbf{h}_t}_{\text{Query}} \right) \underbrace{\mathbf{X}}_{\text{Value}}.$$

The Transformer models a query  $\mathbf{Q}$ , a set of keys  $\mathbf{K}$ , and values  $\mathbf{V}$  individually through three different linear transformations from the input to a Transformer layer.  $\mathbf{Q}$  and  $\mathbf{K}$  are of dimension  $D_K$  and  $\mathbf{V}$  of dimension  $D_V$ . To account for potentially large dot products, the result of the scoring function is further normalized by the root of  $D_K$ ,

$$\mathbf{c}_t = \text{softmax} \left( \frac{\mathbf{K} \cdot \mathbf{Q}_t}{\sqrt{D_K}} \right) \mathbf{V}.$$

The Transformer computes  $h$  of these representation with different *Attention-Heads* which are concatenated and followed by another linear layer. It is of note that due to the loss of recurrence, this formulation can no longer account for positional information and all words would be treated equidistant. To address this problem, the Transformer introduces a *positional encoding* that is added to each word embedding,

$$PE_{(pos,2i)} = \sin \left( pos / 10000^{2i/D_{\text{emb}}} \right)$$

$$PE_{(pos,2i+1)} = \cos \left( pos / 10000^{2i/D_{\text{emb}}} \right).$$

Here,  $pos$  represents the position of the word within a sequence, and  $i$  the dimension within the word embedding vector. Each individual Transformer sublayer, for example the Multi-Head attention, additionally has a residual connection (He et al., 2016), which is a connection that skips the transformation and is followed by a layer normalization (Ba et al., 2016). That means that the output of a sublayer is equal to  $\text{LayerNorm}(x + \text{Sublayer}(x))$ . If the Transformer is used as a decoder, the self-attention only attends to already generated tokens, and the sublayer is followed by another multi-head attention that attends to the input.

The final sublayer of a Transformer layer is an MLP with a ReLU activation function(Hahnloser et al., 2000), defined as

$$\mathbf{MLP}_{\text{Transformer}}(x) = \mathbf{W}^2 \max(0, \mathbf{W}^1 \mathbf{x} + \mathbf{b}^1) + \mathbf{b}^2.$$

## 2.4 APPROXIMATE SEARCH AND TEXT GENERATION

The goal of supervised NLG tasks is to find the sequence from the set of all possible sequences that maximizes the conditional likelihood of the sequence, given the input,

$$\hat{\mathbf{y}} = \arg \max_{\mathbf{y} \in \mathcal{Y}} p(\mathbf{y} | \mathbf{x}).$$

To compute the conditional likelihood, we can apply models that score a sequence, as described in Section 2.2. The standard approach decomposes the probability into individual word-level estimations  $p(y_t | \mathbf{y}_{<t}, \mathbf{x})$ . Neural network-based models using the deep learning building blocks from Section 2.3 can be used to estimate these probabilities.

To use these predictions in a greedy search for  $\hat{\mathbf{y}}$ , one could simply take the highest probability word at each time step. However, it is possible that this choice will lead down a bad path (for instance, first picking the word "an" and then wanting a word starting with a consonant). We, therefore, need to consider predictions that are not the highest-scoring one for each time step, but it is intractable to perform an exact search for  $\hat{\mathbf{y}}$ . Assuming a left-to-right search, we would need to expand  $|\mathcal{V}|$  candidates at each step of the incremental search. That means, if  $\hat{\mathbf{y}}$  has length  $T$ , the search would have to expand  $|\mathcal{V}|^T$  candidates.

A tractable alternative is to use an approximate, heuristic search strategy. The most commonly applied strategy is *beam search*, which is an example of the forward pruning strategy which disregards the majority of expanded candidates without further consideration (Russell and Norvig, 2016). Beam search instead pursues several possible *hypothesis* translations each time step. It does so by building a tree comprising the top  $K$ -hypothesis translations. At each point, all next words are generated for each hypothesis. Of these  $K * |\mathcal{V}|$ , only the most likely  $K$  are retained and the rest is discarded. Once all  $K$  beams have terminated by predicting a special stop token, the final prediction is the translation

with the highest score.

Beam search is the standard approach to generating a sequence of words that is approximately the one with the highest likelihood according to the model. However, this search approach assumes that a single best  $\hat{\mathbf{y}}$  exists. In many NLG cases there exist many different plausible surface realizations that receive a lower likelihood by the model. Being able to actualize these *diverse* predictions is especially crucial if a human has repeated interaction with model-generated text.

If a model can successfully approximate the “real” distribution of language in a given context, sampling is a valid alternative for text generation. Instead of treating the model as a scoring function, we can treat its prediction as a distribution. We sample from it by generating a word  $w$  proportionally to  $p(y_t=w|\mathbf{y}_{}, \mathbf{x})$ .

Often, the distributions resulting from the model have a high entropy, which means that a high share of the probability mass can be spread over incorrect words. It can thus happen that sampling leads to diverse but low-quality samples. To avoid this problem, the distribution can be modulated with a *temperature*, which leads to a lower frequency of low-probability words. In temperature-based sampling, we choose a word  $w$  proportionally to  $p^{1/T}(y_t=w|\mathbf{y}_{}, \mathbf{x})$  for a temperature  $T$ . As we decrease  $T$ , the entropy decreases and words with the highest probabilities are more likely to be chosen.

An alternative approach, not further discussed here is to directly model diverse sequences as part of a model ([Gehrmann et al., 2018a](#)).

*An expression cannot be guaranteed as fully intelligible unless  
an explication or analysis of its meaning has been provided.*

Paul Grice

# 3

## Understanding Users and their Interpretability Needs

A MODEL IS INTERPRETABLE when it can explain its outputs such that users are able to understand its predictions (Doshi-Velez and Kim, 2017, Lipton, 2018). There exist many methods that try to enable the understanding of neural networks. These methods, which have also been called network analysis

or probing methods, aim to derive insights from a fully trained model (Belinkov and Glass, 2019). Since neural architectures are composed of many non-linear building blocks that transform the input data in various ways, their predictions are not inherently explainable. This limitation inhibits their applicability to many use cases where their predictions can have critical consequences, for example in health care. Although a smart assistant to a physician that helps interpret data could bring many benefits, the physician has their own understanding of a problem and their own opinion about the solution. A neural network could either reinforce or contradict this opinion, but since it is not giving any reason for its prediction, the physician should not blindly trust it. If a network could explain its prediction instead, the physician could treat it as an additional input to consider when making their own verdict. Interpretability is thus a crucial goal for the safe and ethical application of all deep learning systems.

As a first step toward interpretable models and predictions, we need to consider *who* an interactive tool is addressing. Therefore, in Chapter 3.1, we introduce a categorization for user types based on how much knowledge a user of a tool has of the underlying model and of the task that it addresses. For example, a clinician has task-specific knowledge, but little model-specific knowledge and is thus classified as *end user*. In contrast, someone who develops new machine learning methods that aim to suggest medical treatments may have limited task-specific knowledge, but is an expert in the model-specific knowledge. We categorize this user as *architect*. The last user group is called the *trainer* which comprises those who use or train models, but don't develop the methodologies. These users often stand in between end users and architects and have some knowledge about both the task and the model.

Once the user type is known to the developer of an interpretability tool, they have to decide on two additional factors that we describe in Chapter 3.2. The first factor differentiates between interactions that aim to help understand a specific prediction from the model, and those that understand the model in general. This decision strongly depends on the needs of the user – do they want to validate that the model has learned expected inductive biases, or do they want an explanation for a prediction?

Furthermore, interfaces for both types of methods can be arbitrarily tightly coupled with the underlying model. Consider an interface that presents a time series of measures to monitor how well a model is training (Wongsuphasawat et al., 2018). This interface does not need to be coupled with the model at all. The only requirement is that the model provides the means to access those statistics. In contrast, consider an interactive tool where model and user take turns developing a video game (Guzdial and Riedl, 2019). The interface needs to enable a user to interact with the input and the output of the model, which requires a tighter coupling.

We exemplify these user types and categories with three case studies and show how interaction can lead to a better intuition and understanding about machine learning models. In NLG, developing an understanding for the machine learning model behavior can be especially challenging, because models take many sequential decisions that all require an explanation. For that reason, we first discuss a simpler case study in Chapter 4, specifically a binary classification task where the output is a zero or a one. We investigate the problem of correctly identifying whether a patient is part of a particular patient cohort based on the textual information from their electronic health records. Traditionally, the most commonly used approach to this problem relies on extracting a number of clinician-defined medical concepts from the text and using machine learning techniques to identify whether a particular patient has a certain condition. We compare the concept extraction-based approach to CNN-based deep learning and show that the CNN can achieve better performance across ten different conditions. Since both approaches rely on information in phrases, either hand-curated or learned, we investigate whether we can use the most predictive phrases in a document as an explanation. For the CNN, we derive the phrase-importance by extending the formulation of the so called first-derivative saliency, which has previously been used to investigate machine translation (Li et al., 2016a). Saliency is a method that targets end users who may know nothing about the model, but have extensive domain knowledge. To evaluate this kind of explanation, we ask clinicians for their preference out of the two explanation types. The results show a clear preference for the CNN-based explanations, but also

point to inherent limitations in the “static” analysis that shows non-interactive explanations.

Humans highly rate static explanations when they can use the explanation to confirm their intuition how a model works. However, when the explanation is not intuitive to the human, it can lead to confusion and less trust. Explanations from interpretability methods thus need to be presented in a way that meshes with the mental model of a person, i.e., similar to the way the person would have given a explanation of their decision (Johnson-Laird, 1983). It is, however, impossible to provide a single explanation that satisfies every person who interacts with model predictions. Moreover, it requires the explicit reflection of the human reasoning process within a model. We, therefore, pose an alternative challenge that can be addressed by combining methods from HCI and visualization with the interpretability methods from machine learning and NLP. We argue that being able to interact with a model allows users to gain an understanding and develop a mental model of the machine learning model behavior. This understanding helps users comprehend how a model operates and to recognize its limitations.

In Chapter 5, we further present a case study of a system that assists trainers and architects investigate the behavior of RNNs. Researchers interested in better understanding RNNs have studied how their hidden state changes over time and noticed some interpretable patterns but also significant noise. We develop LSTMVis, a visual analysis tool for recurrent neural networks with a focus on understanding these hidden state dynamics. With a focus on various language modeling tasks, the tool allows users to test a hypothesis about what global patterns an RNN has learned to detect, for example noun phrases. To test the hypothesis, the tool assists the user in selecting a sequence of model inputs and a set of hidden states for these inputs. The tool matches these states changes to similar patterns in a large data set, and aligns these results with structural annotations from the domain in which the model is operating. We demonstrate several use cases of LSTMVis for analyzing specific hidden state properties on dataset containing nesting, phrase structure, and chord progressions, and demonstrate how the tool can be used to isolate patterns for further statistical analysis. Long-term usage data after

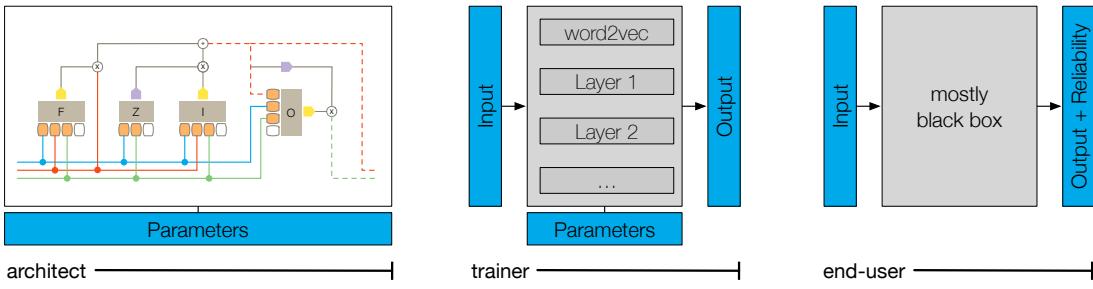
putting the tool online revealed great interest in the machine learning community.

While LSTMVis shows how we can gain insight into patterns that a network learns, it does not help users understand a specific prediction. Sequence-to-sequence models are becoming the standard approach for many real-world NLG applications, most notably translation (Wu et al., 2016). However, these models frequently make mistakes. While these are often harmless, a wrong translation made by the facebook translation system led to a wrongful arrest in the past (Hern, 2017). A first step toward preventing these mistakes is understanding where in a model a specific error originated. To answer this question, we developed a neural debugger, called SEQ2SEQ-Vis, which we present in Chapter 6. SEQ2SEQ-Vis is a visual analysis tool that allows interaction with a trained sequence-to-sequence model through each stage of the translation process. It shows that visualizing the inherently understandable attention within the model can help humans relate some complex model decisions to understandable abstractions. We demonstrate the utility of our tool through several real-world large-scale sequence-to-sequence use cases.

LSTMVis and SEQ2SEQ-Vis demonstrate the need for interactive interfaces that are designed for their respective user types that help people understand and debug the reasoning process of neural networks. LSTMVis shows that relating complex model-internals to understandable examples with similar hidden states can help humans develop an intuition how these states arise. SEQ2SEQ-Vis expands the idea of understandable model internals and demonstrates that inherently understandable parts of a model can be used within an interface to enable an intuitive understanding. Together, the case studies show how interaction with interpretable aspects of a model can be used to understand a model, and that an understandable model-internal reasoning process widens the possible design space for interfaces.

We further observe that the described cases are restricted to the interactions that the model design allows. The more static and non-interpretable the model design is, the more challenging it is to enable interactions with it. We thus conclude that the interpretability is limited by the non-interpretable and

non-interactive model design and that we need inherently interactive and understandable models.



**Figure 3.1:** Views on neural network models for different user roles. The **architect** analyzes and modifies all components of the system. The **trainer** abstracts the model to the main components and parameters and focuses on training on different data sets. The **end user** has the most abstract view on the model and considers whether the output is coherent for a given input.

### 3.1 USER TYPES: ARCHITECTS, TRAINERS, AND END USERS

Presenting clinicians with a list of the most predictive phrases that a model uses to make a prediction allows them to use their domain knowledge to validate whether the explanations match their intuition about what a model has learned. However, a list of predictive phrases may fail to provide enough information when the explanation does not match this intuition or when more relevant phrases exist that model does not consider. One approach to expand on these explanations is to reveal more details about the inner workings of a model. However, this approach requires more domain knowledge about NLP and machine learning which clinicians typically do not have. The solution, therefore, has to strike a balance between the informativeness and difficulty of the explanation.

Since deep neural networks are now widely employed both in the research and industrial setting, methods for the analysis and interpretation are applied and seen by a diverse set of users with different needs. It is crucial to identify the user and their needs before developing and applying interpretability methods for a specific use case. We identified three prototypical user roles, their incentives, and their view on neural networks, which we summarize in Figure 3.1: *architects* develop novel deep learning structures, *trainers* develop new data sets to train existing models, and *end users* apply deep models to

new data.

**ARCHITECTS** are looking to develop new deep learning methodologies or to modify existing deep architectures for new domains. An architect is interested in training and comparing different network structures and comparing how the models capture the features of their domain. We assume that the architects are deeply knowledgeable about machine learning, neural networks, and the internal structure of the system. Their goal is comparing the performance of variant models and understanding the learned properties of the system.

**TRAINERS** are those users interested in applying known architectures to tasks for which they are domain experts. Trainers understand the key concept surrounding deep learning and utilize neural networks as a tool. However, their main focus is on the application domain and utilizing effective methods to solve known problems. Their goal is to use a known network architecture with only minor modifications. They aim to observe how the model performs in the chosen target domain and communicate the results to the domain experts to derive insights from the data. Examples of trainers include a bioinformatician or an applied machine learning engineer.

**END USERS** are the most prevalent role of users. End users utilize networks that were trained by someone else for various tasks. End users do not need to understand the model at all, and are only presented with their outputs. They have a deep domain knowledge and the predictions are aimed to enrich a decision process in which they are involved. Examples for this kind of user are clinicians who are presented with predicted risk scores or bankers who observe default risks on a loan.

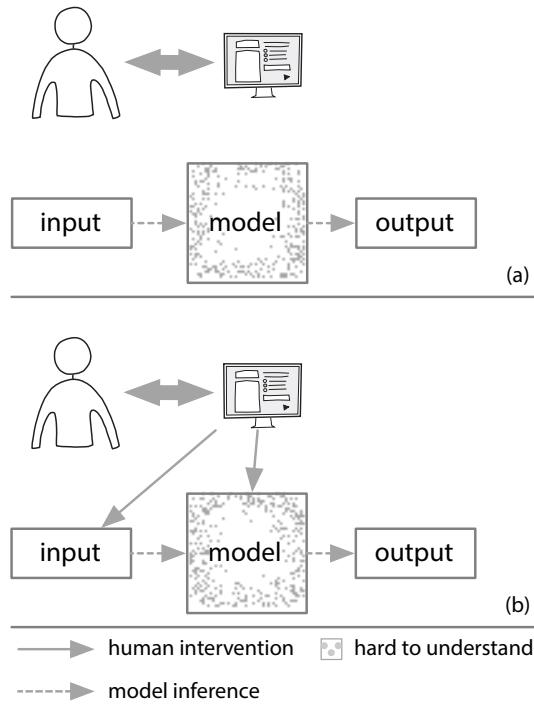
### 3.2 DESIGN SPACE FOR INTEGRATING MACHINE LEARNING INTO INTERFACES

The different user types give insights into the level of model-internal information that an interface can expose and the amount of domain-specific knowledge that the analysis methods can expect. They do not, however, inform us about the tasks that should be completed within the interface and the tightness of the coupling between the interface and the machine learning model. We thus synthesize insights from analyses of the design space for visual interpretation of machine learning models and introduce a classification scheme with a focus on the efforts from both machine learning and visualization experts who co-design interpretability tools (Endert et al., 2017, Liu et al., 2017, Lu et al., 2017, Hohman et al., 2018).

The different co-design approaches are contextualized through a categorization of the design space based on two criteria: (1) the level of integration of a machine learning model and an interactive interface, and (2) applications that aim to understand and shape the model or model-decisions. Table 3.1 shows a categorization of the related work using these criteria. Each of the resulting categories can encompass tools that cater to multiple different user types.

We identify two broad integration approaches between models and visual interfaces (Figure 3.2): *passive observation* and *interactive observation*, which we extend by a third category *interactive collaboration* in chapter 8. As we will discuss in detail in this section, these categories comprise a class of techniques that address different challenges along an analysis pipeline.

In addition to the level of integration and user type, we divide the design space between visual interfaces to *understand the model or the decisions* of the model. Model-understanding describes systems with the goal of understanding the model through its features and parameters. The methods can help to gain insights into or modify the parameters that a model is in the process of learning or has learned already. Decision-understanding systems have the goal of understanding the individual decisions of the model on specific instances. Decision-based applications aim to understand how the model arrives



**Figure 3.2:** An illustration of the two coupling types between interfaces. Interactive observation tools require a tighter coupling and thus enable more interactions than passive observation tools.

at a given output but does not modify the parameter of the model.

We provide a brief overview of the related work for each of the categories in Table 3.1.

### 3.2.1 PASSIVE OBSERVATION

The first stage in our design space is passive observation, in which visualizations present a user with static information about the model or its output. The information can target any user-type and range from global information about model training to heatmaps overlaid over specific inputs. Passive observation interfaces only require a loose coupling between the interface and the model (Figure 3.2(a)).

**MODEL-UNDERSTANDING** Architects and trainers are often concerned about how well a model is training, i.e., the model *performance*. Tools can assist trainers by tracking and comparing different models and metrics, e.g., Tensorboard (Wongsuphasawat et al., 2018). Moreover, it is crucial for trainers to understand whether a model learns a good representation of the data as a secondary effect of the training, and to detect potential biases or origins of errors in a model (Belinkov and Glass, 2019). To address this issue, many model-understanding techniques aim to visualize or analyze learned global features of a model (Carter et al., 2019, Odena et al., 2016, Belinkov et al., 2017a). Most recent work in this direction focuses on visualizing hidden representations of CNNs (Lecun et al., 1998) for computer vision applications. For example, techniques can directly show the images that maximize the activity (Simonyan et al., 2013). For RNNs, Karpathy et al. (2015) use static visualizations to understand hidden states in language models. They demonstrate that selected cells can model clear events such as open parentheses and the start of URLs. In Zeiler and Fergus (2014) use deconvolutional networks to explore the layers of a CNN. This approach is widely used to generate explanations of models, for example, by Yosinski et al. (2015). In Chapter 4, we will explore a similar technique to generate explanations for a CNN that aims to classify text.

**DECISION-UNDERSTANDING** Decision-understanding passive observation tools assist end users in developing a mental model of the machine learning model behavior for particular examples. The most commonly applied decision-understanding techniques present overlays over images (Dey et al., 2015) or texts (Dalvi et al., 2019). There is a rich literature for methods that compute these relevant inputs for specific predictions, for example by computing local decision boundaries or using gradient-based saliency (Li et al., 2016a, Ribeiro et al., 2016, Ross et al., 2017, Zintgraf et al., 2017, Alvarez-Melis and Jaakkola, 2017). Most of these methods focus on classification problems in which only one output exists. Rücklé and Gurevych (2017) address this issue and extend saliency methods to work with multiple outputs in a question-answering system. As an alternative to saliency-methods,

	Model-Understanding	Decision-Understanding
Passive Observation	Activation Atlas ( <a href="#">Carter et al., 2019</a> ), Classification Visualization ( <a href="#">Chae et al., 2017</a> ), DeepEyes ( <a href="#">Pezzotti et al., 2018</a> ), Feature classifiers ( <a href="#">Belinkov et al., 2017a</a> ), Tensorboard ( <a href="#">Wongsuphasawat et al., 2018</a> ), Weight Visualization ( <a href="#">Tzeng and Ma, 2005</a> ), <b>CNN Saliency (Chapter 4, Gehrmann et al., 2018b)</b>	Deconvolution ( <a href="#">Zeiler and Fergus, 2014</a> ), LIME ( <a href="#">Ribeiro et al., 2016</a> ), Neuron Analysis ( <a href="#">Dalvi et al., 2019</a> ), Rationals ( <a href="#">Lei et al., 2016</a> ), RNN Saliency ( <a href="#">Li et al., 2016a</a> ), Structured Interpretation ( <a href="#">Dey et al., 2015</a> ), Prediction Difference ( <a href="#">Zintgraf et al., 2017</a> )
Interactive Observation	ActiVis ( <a href="#">Kahng et al., 2018</a> ), Deep Visualization ( <a href="#">Yosinski et al., 2015</a> ), Embedding Projector ( <a href="#">Smilkov et al., 2016</a> ), GANViz ( <a href="#">Wang et al., 2018a</a> ), Prospector ( <a href="#">Krause et al., 2016</a> ), RetainVis ( <a href="#">Kwon et al., 2019</a> ), RNNVis ( <a href="#">Ming et al., 2017</a> ), ShapeShop ( <a href="#">Hohman et al., 2017</a> ), <b>LSTMVis (Chapter 5, Strobelt et al., 2018b)</b>	Instance-Level Explanations ( <a href="#">Krause et al., 2017</a> ), Manifold ( <a href="#">Zhang et al., 2019a</a> ), NLIZE ( <a href="#">Liu et al., 2019</a> ), RNNBow ( <a href="#">Cashman et al., 2018</a> ), Semantic Image Synthesis ( <a href="#">Chen and Koltun, 2017</a> ), <b>Seq2Seq-Vis (Chapter 6, Strobelt et al., 2019)</b>

**Table 3.1:** A classification of some of the related work, which shows the distinct lack of collaborative interfaces for decision-understanding and shaping. Our CSI framework aims to fill this void, with a particular focus on applications built for end users. However, different user types span all of the previous work, some aiming towards architects, trainers, end users, or a combination of them.

[Ding et al. \(2017\)](#) use a layer-wise relevance propagation technique ([Bach et al., 2015](#)) to understand relevance of input with regard to an output in sequence-to-sentence models. Yet another approach

to understand predictions within text-based models is to find the minimum input that still yields the same prediction (Lei et al., 2016, Li et al., 2016b).

These overlays often represent the activation levels of a neural network for a specific input (Erhan et al., 2009, Le et al., 2012). For example, in image captioning, this method can show a heatmap that indicates which part of an image was relevant to generate a specific word (Xu et al., 2015). Note that often, the same methods can be applied to the model as a whole and to specific instances, which means that those methods could be used for both model and decision-understanding (for example, our case study in Chapter 4).

A commonly noted challenge is the quantitative assessment of these methods. A commonly used assessment of decision-understanding methods may focus on whether generated highlights match human intuition (Kindermans et al., 2019, Adebayo et al., 2018). However, the understanding of a model and its predictions depend on the domain knowledge of the users and the set of investigated examples. The perceived usefulness may thus highly vary between users of a tool. Other applications of decision-understanding methods lie in a pedagogical context where they can be used to develop an understanding of a type of model in general (Hohman et al., 2018, Smilkov et al., 2017a). Here, an evaluation focuses on whether the method helps students understand the concepts behind a model.

### 3.2.2 INTERACTIVE OBSERVATION

Interactive observation interfaces can receive feedback or information from the model itself (Figure 3.2(b)). This feedback enables the testing of multiple hypotheses about the model behavior within a single interface. We classify tools as interactive observation that allow changing inputs or extracting any model-internal representation as part of an interactive analysis. We call these *forward* interactions, analogous to how sending inputs into a model is called the forward pass. This approach can be used by trainers with domain knowledge to verify that a model has learned known structures in the data, or by end users to gain novel insights into a problem. The development of interactive observation methods has

been an active field of research, as summarized in recent review papers (Liu et al., 2017, Lu et al., 2017, Hohman et al., 2018). Interactive observation allows for a richer space of potential interactions than passive observation tools and thus require a closer coupling between visualizations, interface, and the model (Endert et al., 2017).

**MODEL-UNDERSTANDING** In an extension of visualization of learned features, interactive observational tools enable end users and trainers to test hypotheses about global patterns that a model may have learned. One example is Prospector (Krause et al., 2016), which can be used to investigate the importance of features and learned patterns. Alternatively, counterfactual explanations can be used to investigate changes in the outcome of a model for different inputs, thereby increasing trust and interpretability (Wexler et al., 2019, Wachter et al., 2017).

In Chapter 5, we present LSTMVis, an approach to making sense of the complex interactions within a model by relating them to similar training examples. This work is in line with other tools that aim to interactively investigate the hidden representations of RNNs to understand what they have learned Kahng et al. (2018), Ming et al. (2017). Alternative approaches use clustering and dimensionality reduction to visualize the progression of hidden states (Johnson et al., 2017).

**DECISION-UNDERSTANDING** Interactive decision-understanding tools visualize how small changes to an input or the internal representation influence the model prediction. Interactive experimentation can lead to an understanding of a particular prediction by using the local information around only one input (Park et al., 2018). Interactively building this intuition is crucial to end users since past research has shown that statically presenting only a few instances may lead to misrepresentation of the model limitations (Kim et al., 2016a) or the data that the model uses (Vellido et al., 2012). In that line, Olah et al. (2018) build an interactive system that shows that not only the learned filters of a CNN matter, but also their magnitudes.

Moreover, interactive decision-understanding answer counter-factual “what if” questions to understand the robustness of a model to perturbations. [Nguyen et al. \(2015\)](#) show that small perturbations to inputs of an image classifier can drastically change the output. Interactive visualization tools such as Picasso ([Henderson and Rothe, 2017](#)) can manipulate and occlude parts of an image as input to an image classifier. [Krause et al. \(2016\)](#) use partial dependence diagnostics to explain how features affect the global predictions, while users can interactively tweak feature values and see how the prediction responds to instances of interest.

Another desired outcome of interactive observational tools is the testing of hypotheses about local patterns that a model may have learned. As an example, we present `SEQ2SEQ-VIS` in Chapter 6, which allows users to change inputs and constrain the inference of a translation model in order to pinpoint what part of the model is responsible for an error. Similar debugging-focused approaches ([Zhang et al., 2019a](#), [Krause et al., 2017](#)) only visualize the necessary parts of a model to find and explain errors, instead of giving in-depth insights into hidden model states.

*If a lion could talk, we could not understand him.*

Ludwig Wittgenstein

# 4

## Evaluating Explanations in Automated Patient Phenotyping

An area in which it is crucial to deploy understandable and well-functioning predictive models is healthcare. Predictive models are very commonly used in the analysis of electronic health records (EHR). EHRs are the place where caretakers in clinical settings record relevant patient information. Throughout a hospital stay, a patient is continuously monitored and all updates are noted within

their medical record. EHR data comprise both structured data such as International Classification of Diseases (ICD) codes, laboratory results and medications, and unstructured data such as clinician progress notes. While structured data do not require complex processing prior to statistical tests and machine learning tasks, the majority of data exist in unstructured form (Murdoch and Detsky, 2013), since nurses, social workers, and doctors detail most information textual notes. NLP methods can analyze this valuable data, which in conjunction with analyzing structured data can lead to a better understanding of health and diseases (Liao et al., 2015).

One common application of NLP in EHRs is the identification of patient cohorts in secondary analysis, a task that is also called patient phenotyping. In secondary analysis, researchers aim to find patient populations with certain commonalities, for example advanced lung disease, within all the patient records in the EHR. Studying these populations can give valuable insights into the disease progression, identify preventable hospital admission patterns, and even assist in the development of new treatment strategies (Ananthakrishnan et al., 2013, Pivovarov and Elhadad, 2015, Halpern et al., 2016). NLP is required in this task, since the information is often only found in the textual notes. NLP-based secondary analysis of data from EHRs is thus crucial to better understand the heterogeneity of treatment effects and to individualize patient care (Celi et al., 2016).

A popular approach to patient phenotyping using NLP is based on extracting medical phrases from texts and using them as input to build a predictive model (Hripcsak and Albers, 2013). The dictionary of relevant phrases is often task-specific and its development requires significant effort and a deep understanding of the task from domain experts (Carrell et al., 2014). A different approach is to develop a fully rule-based algorithm for each condition (Kirby et al., 2016). While this approach leads to a high performance, the development of these phenotyping models is time-consuming due to the laborious tasks required of clinicians and are thus rarely deployed outside of the research area. However, deep learning might prove to be a generalizable approach for phenotyping with less intense domain expert involvement. Applications of deep learning for other tasks in healthcare have

shown promising results; examples include mortality prediction (Ranganath et al., 2016), patient note de-identification (Dernoncourt et al., 2017), skin cancer detection (Esteva et al., 2017), and diabetic retinopathy detection (Gulshan et al., 2016).

However, in order for deep learning to be a valid alternative to the clinician-supervised processing pipelines, their predictions need to be understandable. This is crucial for healthcare applications since results can directly impact decisions about patients health. Furthermore, clinicians are specialized domain experts and thus expect applications to support their decision making as opposed to make decisions for them. Therefore, interpretable models are required so that clinicians can trust and control their results (Caruana et al., 2015). Interpretable algorithms are even required by law in the European Union due to the “right to explanation” (Goodman and Flaxman, 2017). While much work has been done to understand deep learning NLP models and develop understandable models (e.g., Zeiler et al., 2010, Yosinski et al., 2015, Strobelt et al., 2018b), their complex interactions between inputs are inherently less understandable than parameters of linear models or rule-based approaches that are fully clinician-defined.

We aim to better understand the differences between the deep learning and concept extraction approaches in terms of performance and interpretability. Specifically, we assess the utility of CNNs as an approach to text-based patient phenotyping. We compare CNNs to entity extraction systems using the Mayo clinical Text Analysis and Knowledge Extraction System (cTAKES, Savova et al., 2010), and simple baseline methods such as logistic regression models using n-gram features. Using a corpus of 1,610 discharge summaries that were annotated for ten different phenotypes, we show that CNNs outperform both extraction-based and n-gram-based methods. To evaluate the interpretability of each model, we follow the identified properties of the end user from Section 3.1 in that we aim to abstract away anything network-specific and instead report the most important input features for a predictions. We assess whether the identified relevant phrases are correctly associated with each phenotype and ask clinicians for their preferred explanation of a prediction. The results indicate that the

explanations from the CNN-based models are often preferred to the more traditional models, but are often noisy. Moreover, clinicians remarked that, while these explanations help them understand the prediction, they do not always cover the same phrases they would have used to support their assessment.

#### 4.1 PHENOTYPING IN LARGE EHR DATASETS

With the growing adoption rate of EHRs (Charles et al., 2013), researchers gain access to rich data sets, such as the Medical Information Mart for Intensive Care (MIMIC) database (Saeed et al., 2002, Johnson et al., 2016), and the Informatics for Integrating Biology and the Bedside (i2b2) datamarts (Uzuner et al., 2008, Uzuner, 2009, Uzuner et al., 2010, 2011, Sun et al., 2013, Stubbs and Uzuner, 2015). These data sets can be explored and mined in numerous ways (Jensen et al., 2012). Intelligent applications for patient phenotyping can support clinicians by reducing the time they spend on chart reviews, which takes up a significant fraction of their daily workflow (Chen et al., 2016, Topaz et al., 2016).

Accurate patient phenotyping is required for secondary analysis of EHRs to correctly identify the patient cohort and to better identify the clinical context (Ackerman et al., 2016, Razavian et al., 2015). Studies employing a manual chart review process for patient phenotyping are naturally limited to a small number of preselected patients. Therefore, NLP is necessary to identify information that is contained in text but may be inconsistently captured in the structured data, such as recurrence in cancer (Carrell et al., 2014, Strauss et al., 2013), whether a patient smokes (Uzuner et al., 2008), classification within the autism spectrum (Lingren et al., 2016), or drug treatment patterns (Savova et al., 2012). However, unstructured data in EHRs, for example progress notes or discharge summaries, are not typically amenable to simple text searches because of spelling mistakes, synonyms, and ambiguous terms (Nadkarni et al., 2011). To help address these issues, researchers utilize dictionaries and ontolo-

gies for medical terminologies such as the unified medical language system (UMLS) (Bodenreider, 2004) and the systematized nomenclature of medicine – clinical terms (SNOMED CT) (Spackman et al., 1997).

Examples of systems that employ such databases and extract concepts from text are the KnowledgeMap Concept Identifier (KMCI) (Denny et al., 2003), MetaMap (Aronson and Lang, 2010), Medlee (Friedman et al., 1994), MedEx (Xu et al., 2010), and the cTAKES. These systems identify phrases within a text that correspond to medical entities (Savova et al., 2010, Denny et al., 2012). This significantly reduces the work required from researchers, who previously had to develop task-specific extractors (Hripcsak et al., 2002). Extracted entities are typically filtered to only include concepts related to the patient phenotype under investigation and either used as features for a model that predicts whether the patient fits the phenotype, or as input for rule-based algorithms (Hripcsak and Albers, 2013, Lingren et al., 2016, Pradhan et al., 2015). Liao et al. (Liao et al., 2015) describe the process of extraction, rule-generation and prediction as the general approach to patient phenotyping using the cTAKES (Ananthakrishnan et al., 2013, Perlis et al., 2012, Xia et al., 2013, Liao et al., 2010), and test this approach on various data sets (Carroll et al., 2012). The role of clinicians in this task is both to annotate data and to develop a task-specific dictionary of phrases that are relevant to a patient phenotype. Improving existing approaches often requires significant additional time-investment from the clinicians, for example by developing and combining two separate phrase-dictionaries for pathology documents and clinical documents (Carrell et al., 2014). The cost and time required to develop these algorithms limit their applicability to large or repeated tasks. While a usable system would offset the development costs, it does not address the problem that a specialized NLP system would have to be developed for every task in a hospital. Moreover, algorithms often do not transfer well between different hospitals, warranting extensive transferability studies (Ye et al., 2017). The deep learning based approach we evaluate in this paper does not require any hand-crafted input and can easily be retrained with new data. This can potentially increase the transferability of studies while removing the time

required to develop new phrase-dictionaries.

#### 4.2 DATA

The notes used in this study are extracted from the MIMIC-III database (Johnson et al., 2016). MIMIC-III contains de-identified clinical data of over 53,000 hospital admissions for adult patients to the intensive care units (ICU) at the Beth Israel Deaconess Medical Center from 2001 to 2012. The dataset comprises several types of clinical notes, including discharge summaries ( $n=52,746$ ) and nursing notes ( $n=812,128$ ). We focus on the discharge summaries since they are the most informative for patient phenotyping (Sarmiento and Dernoncourt, 2016). More specifically, we investigate phenotypes that may associate a patient with being a ‘frequent flyer’ in the ICU (defined as  $\geq 3$  ICU visits within 365 days). As many as one third of readmissions have been suggested to be preventable; identifying modifiable risk factors is a crucial step to reducing them (Kocher and Adashi, 2011). However, the underlying conditions that lead to frequent hospital visits are often not captured in the structured disease codes, which motivates the use of NLP techniques to identify them for further study.

To create a balanced dataset, we extracted the discharge summary of the first visit from all 415 ICU frequent flyers in MIMIC-III, as well as 313 randomly selected summaries from later visits of the same patients. We additionally selected 882 random summaries from patients who are not frequent flyers, yielding a total of 1,610 notes. The cTAKES output for these notes contains a total of 11,094 unique CUIs.

Following clinician-defined definitions, 1,610 notes were annotated for ten phenotypes: advanced / metastatic cancer, advanced heart disease, advanced lung disease, alcohol abuse, chronic pain, chronic neurologic dystrophies, depression, obesity, substance abuse, other psychiatric disorders. To ensure high-quality labels and minimize errors, each note was labeled at least twice for each phenotype. The annotators included two clinical researchers, two junior medical residents, two senior medical resi-

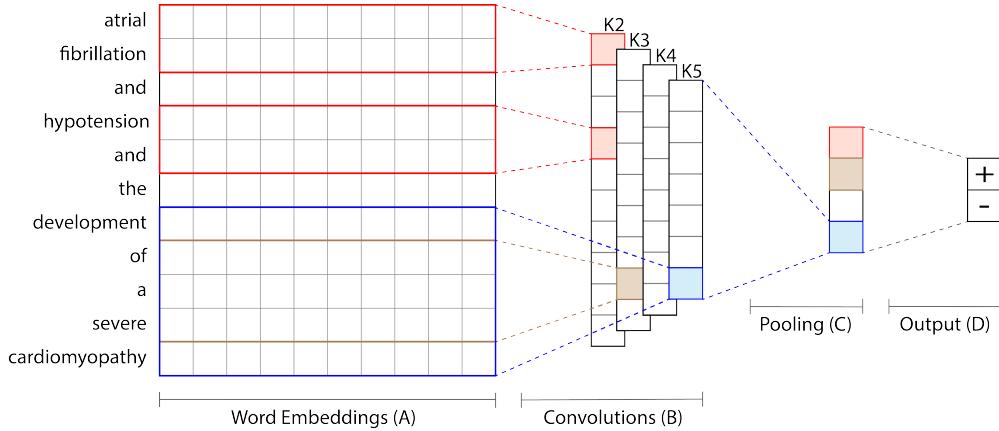
dents, and a practicing intensive care medicine physician. In the case that the annotators were uncertain or disagreed, one of the senior clinicians decided on the final label. The resulting frequency of each phenotype varies from 126 to 460 cases, which corresponds to between 7.5% and 28.6% of the dataset. When measuring the inter-annotator agreement through the Cohen’s Kappa measure well specified phenotypes such as depression have a very high agreement (0.95), whereas other phenotypes, such as chronic neurologic dystrophies, have a lower agreement 0.71 and required more interventions by senior clinicians.

### 4.3 METHODS

#### 4.3.1 CONCEPT-EXTRACTION BASED METHODS

We use cTAKES to extract concepts from each note. In a first step, cTAKES tokenizes the notes and normalizes the tokens such that tokens with variations (e.g. plural) are represented as their base form. The normalized tokens are tagged for their part-of-speech (e.g. noun, verb), and a shallow parse tree is constructed to represent the grammatical structure of a sentence. Finally, a named-entity recognition algorithm uses this information to detect named entities for which a concept unique identifier (CUI) exists in UMLS (Spackman et al., 1997).

We follow previous phenotyping work by using the frequency of occurrences of relevant concepts in a note as input to machine learning algorithms to directly learn to predict a phenotype (Liao et al., 2015, Bates et al., 2016, Kang et al., 2013). We specify two different approaches to using the cTAKES output. The first approach uses the complete list of extracted CUIs, represented as a bag-of-CUIs as input to further processing steps. In the second approach, clinicians specify a dictionary comprising all clinical concepts that are relevant to the desired phenotype (e.g. Alcohol Abuse) as described by Carrell et al. (Carrell et al., 2014). Due to the fact that cTAKES detects negations, the occurrences of negated and non-negated CUIs are counted separately.



**Figure 4.1: An overview of the CNN architecture.** (A) Each word within a discharge note is represented as its word embedding. (B) Convolutions of different widths are applied to word sequences of the corresponding length. (C) The resulting vectors are reduced to only the highest value for each of the different convolutions via a pooling operation. (D) The final prediction is made by computing a weighted combination of the pooled values and applying a sigmoid function. This figure is adapted with permission from [Kim \(2014\)](#).

In the predictive model, the input representation is first transformed using the term frequency-inverse document frequency (TF-IDF), following [Halpern et al. \(2016\)](#). We then train multiple predictive non-neural models to detect whether a patient has a condition. For an accurate comparison to approaches in literature, we train a random forest (RF), a naive Bayes (NB), and a logistic regression (LR) model.

#### 4.3.2 CONVOLUTIONAL NEURAL NETWORKS

We compare the concept-extraction baseline to a text-classification CNN as described in Chapter 2.3. For each of the phenotypes, we train a separate CNN that uses the text in a note  $\mathbf{x}$  to predict the probability that the patient has a phenotype,  $y$ . The text in a discharge summary is first represented as a sequence of word embeddings  $\mathbf{x}_1, \dots, \mathbf{x}_S$ . Word embeddings have shown to improve performance on other tasks based on EHRs, for example named-entity recognition, and avoid having to create a database of common synonyms and misspellings ([Carrell et al., 2014](#), [Wu et al., 2015](#)). They can be

pre-trained on a larger corpus of texts, which improves results of the NLP system and reduces the amount of data required to train a model (Erhan et al., 2010, Luan et al., 2015). We thus pre-train our embeddings with word2vec (Mikolov et al., 2013) on all discharge notes available in the MIMIC-III database (Johnson et al., 2016).

In parallel, we apply convolutional filters with different widths  $1, \dots, K$  to the sequence of word embeddings. For each filter width  $k$ , this operation results in a sequence of hidden representations  $\mathbf{h}_1^{(k)}, \dots, \mathbf{h}_{S-k+1}^{(k)}$ . We apply max-over-time-pooling to each of these sequences individually, which means that we extract the highest values for each dimension for each of the filters (Collobert et al., 2011). We are left with  $K$  pooled vectors, each of dimension  $D$ ,  $\hat{\mathbf{h}}^{(1)}, \dots, \hat{\mathbf{h}}^{(K)}$ . In a final step, we concatenate the pooled vectors and apply a final linear layer, such that

$$y = \sigma(\mathbf{W} \left[ \hat{\mathbf{h}}^{(1)}, \dots, \hat{\mathbf{h}}^{(K)} \right] + b).$$

An overview of our architecture is shown in Fig 4.1.

#### 4.3.3 ADDITIONAL BASELINES

We further investigate a number of baselines to compare with the more complex approaches. We start with a bag-of-words based logistic regression and gradually increase the complexity of the baselines until we reach the CNN. This provides an overview of how adding more complex features impact the performance in this task. Moreover, this investigation shows what factors contribute most to the CNN performance.

**BAG OF WORDS** The simplest possible representation for a note is a bag of words (BoW) which counts phrases of length 1. Let  $\mathcal{V}$  denote the vocabulary of all words, and  $v_i$  the word at position  $i$ . Let further  $\delta(v_i) \in \mathbb{R}^{1 \times |\mathcal{V}|}$  be a one-hot vector with a one at position  $v_i$ . Then, a note  $\mathbf{x}$  is represented

as  $\mathbf{x}_{BoW} = \sum_i \delta(v_i)$ . A prediction is made by computing  $y = \sigma(\mathbf{W}\mathbf{x}_{BoW} + b)$ , where  $\mathbf{W}$  and  $b$  are trainable parameters.

**N-GRAMS** The bag-of-words approach can be extended to include representations for longer phrases, also called n-grams, as well. Consider a note “The sick patient”. While a bag-of-words approach considers each word separately, a two-gram model additionally considers all possible phrases of length two. Thus, the model would also consider the phrases “The sick” and “sick patient”. Since the number of possible phrases grows exponentially in the size of the vocabulary, the data for longer phrases becomes very sparse and the n-gram size can’t be increased too much. In this study, we present results for models with phrase lengths of up to 5.

#### 4.4 DERIVING SALIENT FEATURES

The inability of humans to understand predictions of complex machine learning models poses a difficult challenge when such approaches are used in healthcare (Lipton, 2018). Crucially, well-performing approaches should be understood and trusted by those who use them. Users of these approaches should be aware of biases that models learn to avoid basing medical decisions on them. One example of bias was found in mortality prediction pneumonia patients where an occurrence of asthma seemingly increased the survival probability of a patient (Cooper et al., 1997, Caruana et al., 2015). This result was due to an institutional practice of admitting all patients with pneumonia and a history of asthma to the ICU regardless of disease severity. As a result, a history of asthma was strongly correlated with a lower illness severity. If someone blindly interpreted the result of this analysis as actionable, patients with asthma could have been assigned a lower priority due to an underestimate of their mortality risk. This correlation was only detected because the model was linear which means that the model weights associated with a feature such as asthma were directly interpretable. Experts can double-check whether the features with highest and lowest correspond to their domain knowledge.

We approach the interpretability of the cTAKES and CNN model in a similar way. To measure the interpretability of each approach, clinicians are presented a list of phrases or concepts that are most salient for a particular prediction; this list either manifests as an actual list or as highlights of the original text (Lei et al., 2016, Ribeiro et al., 2016).

We first consider the filtered cTAKES random forest approach. The filtering of the input-features ensures that all features directly pertain to the specific phenotype (Ranganath et al., 2016). We rank the importance of each remaining CUI using the gini importance of the trained model (Stone, 1984). The resulting ranking is a direct indication of the globally most relevant CUIs. An individual document can be analyzed by ranking only the importance of CUIs that occur in this document.

To compute the most relevant phrases for the CNN, we propose a modified version of the first-derivative saliency as defined by Li et al. (2016a). They define the saliency for a neural network as the norm of the gradient of the loss function for a particular prediction with respect to an input  $\mathbf{x}$ . Let  $S_1$  denote the Saliency function for the prediction 1, which in our case corresponds to a positive identification with a phenotype. Then, the saliency for a word  $\mathbf{x}_i$  is defined as

$$S_1(\mathbf{x}_i) = \left\| \frac{\partial L(1, y)}{\partial \mathbf{x}_i} \right\|_2.$$

We are not able to directly apply this method for the CNN, since  $\mathbf{x}_i$  only represents a single word or a single dimension of its embedding. Thus, we extend the saliency definition to instead compute the relevance of entire phrases by taking the gradient with respect to the pooled features, a measure which we call phrase-saliency. The phrase-saliency approximates how much a phrase contributes to a prediction,

$$S_1(\mathbf{x}_{i:i+k-1}) = \left\| \frac{\partial L(1, y)}{\partial \hat{\mathbf{h}}_i^{(k)}} \right\|_2.$$

Since we employ multiple feature maps of different widths, we compute the phrase-saliency across

all of them, and use those with maximum value. The saliency can be measured on a per-document basis. To arrive at the globally most relevant phrases, we iterate over all documents and measure the phrases that had the highest phrase-saliency while removing duplicate phrases. Alternative methods not considered in this work search the local space around an input (Ribeiro et al., 2016), or compute a layer-wise backpropagation (Denil et al., 2014, Arras et al., 2017, Bach et al., 2015, Arras et al., 2016).

We note that it was recently pointed out by Smilkov et al. (2017b) that our definition of saliency is limited when the model performs too well. Specifically, as the loss converges toward 0, the magnitude of the gradients decreases until the saliency can no longer be measured. They thus develop smoothgrad, a method that is robust to this effect by first introducing noise around an input and then repeatedly computing the saliency for those samples. Koehn (2019) further show that the method can lead to strong estimates of the saliency for NLP models.

## 4.5 EVALUATION

### 4.5.1 QUANTITATIVE PERFORMANCE MEASURES

We evaluate the precision, recall, F1-score, and area under the ROC curve (AUC) of all models as a quantitative measure. The F-score is derived from the confusion matrix for the results on the test set. A confusion matrix contains four counts: true positive (TP), false positive (FP), true negative (TN), and false negative (FN). The precision  $P$  is the fraction of correct predictions out of all the samples that were predicted to be positive  $\frac{TP}{TP+FP}$ . The recall  $R$  is the percentage of true positive predictions in relation to all the predictions that should have been predicted as positive  $\frac{TP}{TP+FN}$ . The F1-score is the harmonic mean of both precision and recall  $2 * \frac{P*R}{P+R}$ .

For all models, the data is randomly split into a training, validation, and test set. 70% of the labeled data is used as the training set, 10% as validation set and 20% as test set. While splitting, we ensure that a patients' notes stay within the set, so that all discharge notes in the test set are from patients

that were not previously seen by the model. The reported numbers across different models for the same phenotype are obtained from testing on the same test set. The validation set is used to choose the hyperparameters for the models.

In summary, we present results for the following approaches:

**CNN** The convolutional neural network with best performing convolution width

**BoW** Baseline using a bag of words representation of a note and logistic regression

**n-gram** Baseline using an n-gram representation of a note and logistic regression

**cTAKES full** The best performing model that uses the full output from cTAKES

**cTAKES filter** The best performing model using the filtered CUI-list from cTAKES

#### 4.5.2 ASSESSING INTERPRETABILITY

In order to evaluate how understandable the predictions of the different approaches are, we conducted a study of the globally most relevant phrases and CUIs. For each phenotype, we computed the five most relevant features, yielding a total of 50 phrases and 50 CUIs. We then asked clinicians to rate the features on a scale from 0 to 3 with the following descriptions for each rating:

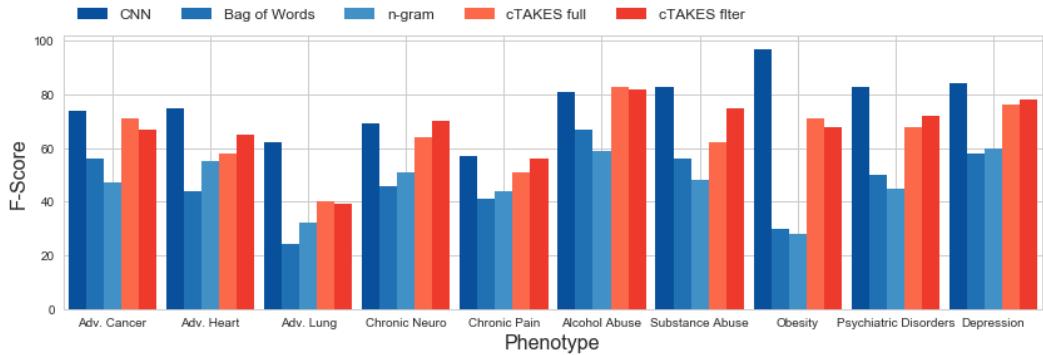
- 0** The phrase/CUI is unrelated to the phenotype.
- 1** The phrase is associated with the concept subjectively from clinical experience, but is not directly related (e.g. alcohol abuse for psychiatric disorder).
- 2** The phrase has to do with the concept, but is not a definite indicator of its existence (e.g. a medication).
- 3** The phrase is a direct indicator of the concept or very relevant (e.g. history of COPD for advanced lung disease).

The features were shown without context other than the name of the phenotype. We additionally provided an option to enter free text comments for each phenotype. We note that all participating clinicians were involved with annotation of the notes and are aware of the definitions for the phenotypes. However, they were not told about the origin of the phrases before rating them in order to prevent potential bias. In total, we collected 300 ratings, an average of three per feature.

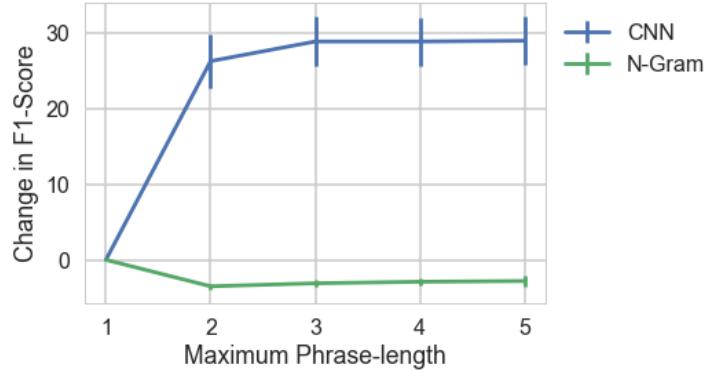
#### 4.6 RESULTS

We show an overview of the F1-scores for different models and phenotypes in Fig 4.2. For almost all phenotypes, the CNN outperforms all other approaches. For some of the phenotypes such as Obesity and Psychiatric Disorders, the CNN outperforms the other models by a large margin. A  $\chi^2$  test confirms that the CNN's improvements over both the filtered and the full cTAKES models are statistically significant at a  $p \leq 0.01$  level. There is only a minimal improvement when using the filtered cTAKES model, which requires much more effort from clinicians, over the full cTAKES model. The  $\chi^2$  test confirms that there is no statistically significant improvement of this method on our data with a  $p$ -value of 0.86. We also note that the TF-IDF transformation of the CUIs yielded a small average improvement in AUC of 0.02 ( $\sigma = 0.03$ ) over all the considered models.

The n-gram and bag-of-words based methods are consistently weaker than the CNN, corroborating the findings in literature that word embeddings improve performance of clinical NLP tasks (Wu et al., 2015). We additionally investigate whether considering longer phrases improves model performance. In Fig 4.3, we show the difference in F1-score between models with phrases up to a certain length and models that use bag-of-words or bag-of-embeddings. There is no significant difference in performance for longer phrases in n-gram models. There is, however, a significant improvement for phrases longer than one word for the CNN, showing that the CNN model architecture complements the embedding-based approach and contributes to the result of the model.



**Figure 4.2: Comparison of achieved F1-scores across all tested phenotypes.** The left three models directly classify from text, the right two models are concept-extraction based. The CNN outperforms the other models on most tasks.



**Figure 4.3: Impact of phrase length on model performance.** The figure shows the change in F1-score between a model that considers only single words and a model that phrases up to a length of 5.

Experiments that used both raw text and CUIs as input to a CNN showed no improvement over only using the text as input. This shows that the information encoded in the CUIs is already available in the text and is detected by the CNN. We hypothesize that encoding information available in UMLS beyond the CUI itself can help to improve the phenotype detection in future work.

We show the most salient phrases according to the CNN and the filtered cTAKES LR models for Advanced Heart Disease and for Alcohol Abuse in Table 4.1. Both tables contain many of the phrases mentioned in the annotation instructions as highly relevant to a phenotype, such as “Cardiomyopa-

thy” for advanced heart disease. We also observe mentions of “CHF” and “CABG” for Advanced Heart Disease for both models, which are common medical conditions associated with advanced heart disease, but were not sufficient requirements on their own according to the annotation scheme. The model still learned to associate those phrases with advanced heart disease, since those phrases also occur in many notes from patients that were labeled positive for advanced heart failure. The phrases for Alcohol Abuse illustrate how the CNN can detect mentions of the condition in many forms. Without human input, the CNN learned that EtOH and alcohol are used synonymously with different spellings and thus detects phrases containing either of them. The filtered cTAKES RF model surprisingly ranks victim of abuse higher than the direct mention of alcohol abuse in a note, and finds that it is very indicative of Alcohol Abuse if an ethanol measurement was taken. While the CUIs extracted by cTAKES can be very generic, such as “Atrium, Heart” or “Heart”, the salient CNN phrases are more specific.

In the quantitative study of relevant phrases and CUIs, phrases from the CNN received an average rating of 2.44 ( $\sigma=0.89$ ), and the cTAKES based approach received an average rating of 1.9 ( $\sigma=0.97$ ). A t-test for two independent samples showed that there is a statistically significant difference between the two with  $p < 0.00001$ . This indicates that the five most relevant features from the CNN are more relevant to a phenotype than the five most relevant features from a cTAKES-based model. The free-text comments confirm our descriptive results; the CNN-based phrases are seen as specific and directly relating to a patient’s condition while the CUI’s are seen as more generic. Moreover, clinicians were impressed to see phrases such as “h o withdrawal” for alcohol abuse (short for “history of”), which are typically difficult to interpret by non-experts. Some of the longer phrases from the CNN for the depression phenotype showed the word “depression” amids other diseases, indicating that the phrase is taken from a diagnosis section of the discharge summary. Clinicians commented that this helped them to contextualize the phrase and that it was more helpful than seeing the word “depression” in isolation. This indicates that giving further contextual information can help to increase understanding

cTAKES	CNN
<b>Advanced Heart Disease</b>	
Magnesium	Wall Hypokinesis
Cardiomyopathy	Port pacer
Hypokinesia	Ventricular hypokinesis
Heart Failure	p AVR
Acetylsalicylic Acid	post ICD
Atrium, Heart	status post ICD
Coronary Disease	EF 20 30
Atrial Fibrillation	bifurcation aneurysm clipping
Coronary Artery	CHF with EF
Disease	cardiomyopathy , EF 15
Aortocoronary Bypasses	( EF 20 30
Fibrillation	coronary artery bypass graft
Heart	respiratory viral infection by DFA
Catheterization	severe global free wall hypokinesis
Chest	Class II , EF 20
Artery	lateral CHF with EF 30
CAT Scans, X-Ray	anterior and atypical hypokinesis akinesis
Hypertension	severe global left ventricular hypokinesis
Creatinine Measurement	's cardiomyopathy , EF 15
<b>Alcohol Abuse</b>	
Victim of abuse	Consciousness Alert
Ethanol Measurement	Alcohol Abuse
Alcohol Abuse	EtOH abuse
Thiamine	Alcoholic Dilated
Social and personal history	ETOH cirrhosis
Family history	heavy alcohol abuse
Hypertension	evening Alcohol abuse
Injuries risk	Drug Reactions Attending
Pain	alcohol withdrawal compartment syndrome
Sodium	EtOH abuse with multiple
Potassium Measurement	liver secondary to alcohol abuse
Plasma Glucose Measurement	abuse crack cocaine, EtOH

**Table 4.1:** The most salient phrases for Advanced Heart Failure and Alcohol Abuse. The salient cTAKES CUIs are extracted from the filtered RF model.

of predictions.

#### 4.7 DISCUSSION AND CONCLUSION

Our results show that CNNs provide a valid alternative approach to the identification of patient conditions from text. However, we notice a strong variation in the results between phenotypes with AUCs between 73 and 100, and F1-scores between 57 and 97, even with consistent annotation schemes and high annotator agreement. Some concepts such as Chronic Pain are especially challenging to detect, even with 321 positive examples in the data set. This makes it difficult to compare our results to other reported metrics in the literature, since studies typically consider different concepts for detection. This problem is further amplified by the sparsity of available studies that investigate unstructured data (Liao et al., 2015), and the lack of standardized datasets for this task. We hope that the release of our annotations will support work towards a more comparable performance in text-based phenotyping.

Since bias in data collection and analysis is at times unavoidable, models are required to be interpretable in order for clinicians to be able to detect such biases, and alter the model accordingly (Caruana et al., 2015). Furthermore, interpretable models lead to an increased trust from the people who use them (Lipton, 2018). The interpretability is typically considered to be a major advantage of rule or concept-extraction based models that are specifically tailored to a given problem. Clinicians have full control over the dictated phrases that are used as input to a model. We demonstrated that CNNs can be interpreted in the same way as concept-extraction based models by computing the saliency of inputs. This even leads to a higher level of interpretability in that the extracted phrases are more relevant to a phenotype than the extracted CUIs. However, a disadvantage of CNNs is that they are designed to consider more different phrases than concept-extraction based methods. Thus, lists of salient phrases will naturally contain more items, making it more difficult to investigate which phrases lead to a prediction. However, restricting the list to a small number of items with the highest saliency coefficients or only including those above a saliency threshold can compensate for the length of the list. The ques-

tion that developers of such models will ultimately have to answer is whether the trade-off between increased performance is worth the additional effort to extract and show relevant phrases.

Clinicians further noted the general limitation of saliency methods that they only provide static and superficial insight into a prediction. Many clinicians noted that seeing the relevant phrases as highlights within a full note would be much more useful than seeing them in isolation, which shows that there are many opportunities for smart interfaces that help clinicians parse medical notes. Moreover, we experienced that some of the phrases started discussions and sparked hypotheses about a patient whenever the explanation did not match the expectation of a clinician. Having more powerful methods and interactions would allow clinicians to follow these hypotheses.

Taking all these points into consideration, we conclude that deep learning provides a valid alternative to concept extraction based methods for patient phenotyping. Using CNNs can significantly improve the accuracy of patient phenotyping without needing any phrase-dictionary as input. We showed that concerns about the interpretability of deep learning can be addressed by computing a gradient-based saliency to identify phrases associated with different phenotypes. We propose that CNNs should be employed alongside concept-extraction based methods to analyze and mine unstructured clinical narratives and augment the structured data in secondary analysis of health records. However, we also identified opportunities for future development of methods and interactive interfaces that help with the understanding of complex neural models.

# 5

## Interactively Understanding Recurrent Neural Networks

The saliency methods introduced above can identify features that are most relevant for a particular prediction. However, models for NLG have to make multiple sequential predictions, each conditioned on all the previous time steps. Thus, the analysis of these models requires different methods. In addition, we identified that non-interactive explanations may not be sufficient to fully understand

the inductive biases of a model. In this section, we aim to explore what a model has learned from the perspective of architects and trainers. This definition allows us to develop interpretability methods that involve knowledge about the underlying model which can lead to more powerful interactions.

Recall from Chapter 2.3.4 that most commonly used approach to sequence modeling are RNNs. RNNs produce a time-series of hidden states in  $\mathbb{R}^D$  that change as the network observes inputs. A function **RNN** uses an input representation  $\mathbf{x}_t$  at a time step  $t$  and its previous hidden state  $\mathbf{h}_{t-1}$  to compute the new hidden state  $\mathbf{h}_t$ . RNNs utilize these hidden states to represent the features of sequence of inputs. This work mainly focuses on the RNN-variant of long short-term memory networks (LSTM, Hochreiter and Schmidhuber, 1997). LSTMs maintain a cell state vector  $\mathbf{c}_t$  in addition to the hidden state vector at each time step. For simplicity, however, we refer to any hidden representation within a RNN as *hidden states*.

Previous empirical results indicate that these hidden states learn to capture complex relationships between the words within a sentence or document. Among others, recurrent representations have led directly to improvements in machine translation (Kalchbrenner and Blunsom, 2013, Sutskever et al., 2014), speech recognition (Amodei et al., 2016), music generation (Boulanger-Lewandowski et al., 2012), and text classification (Dai and Le, 2015). The application we investigate here is language modeling (Mikolov et al., 2010, Zaremba et al., 2014). In language modeling, at time  $t$  the prefix of words  $x_1, \dots, x_t$  is taken as input and the goal is to model the distribution over the next word  $p(x_{t+1}|x_1, \dots, x_t)$ . An RNN is used to produce the corresponding hidden representation  $\mathbf{h}_t$ .  $x_{t+1}$  is then predicted by applying a multi-class classification layer based on the hidden feature-state vector  $\mathbf{h}_t$ , such that  $p(x_{t+1}|x_1, \dots, x_t) = \text{softmax}(\mathbf{W}\mathbf{h}_t + \mathbf{b})$ , where  $\mathbf{W}, \mathbf{b}$  are parameters.

While RNNs have shown clear improvements for sequence modeling, it has proven very difficult to interpret their feature representation. As such, it remains unclear exactly *how* a particular model is representing long-distance relationships within a sequence. Typically, RNNs contain millions of parameters and utilize repeated transformations of large hidden representations under time-varying

conditions. How do we enable users to explore complex network interactions in an RNN and directly connect these abstract representations to human understandable inputs?

In this work, we aim to analyze the change in the hidden representations over time. We refer to the representation changes as the *hidden state dynamics* produced by the model. Specifically, we focus on changes in a subset of the values in  $\mathbf{h}_t$ . We are interested in each  $d \in \{1 \dots D\}$  and in particular the change of a single hidden state  $h_{t,d}$  as  $t$  varies.

To enable the visual analysis of the hidden state dynamics, we have developed LSTMVis. LSTMVis enables its users to form hypotheses about what an RNN may have learned and allows the exploration of the hidden state dynamics as the means to investigate the hypotheses. The investigation of the hidden states requires knowledge about the underlying model, an assumption we did not make with the saliency in the medical domain. Due to this, LSTMVis is focused on architects and trainers, for which we performed a goal and task analysis to develop effective visual encodings and interactions. Our system can be used to analyze any kind of hidden state within recurrent networks, even the LSTM gates, however, in our experiments we found that the cell states are most indicative of what a network has learned. LSTMVis combines a time-series based *select* interface with an interactive *match* tool to search for similar hidden state patterns in a large dataset<sup>1</sup>.

## 5.1 VISUALIZATION FOR UNDERSTANDING NEURAL NETWORKS

Our core contribution, visualizing the state dynamics of RNNs in a structured way, is inspired by previous work on convolutional neural networks in vision applications (Simonyan et al., 2013, Zeiler and Fergus, 2014). In linguistic tasks, visualizations have shown to be useful tool for understanding certain aspects of RNNs. Karpathy et al. (2015) use static visualization techniques to help understand hidden states in language models. Their work demonstrates that selected cells can model clear events

---

<sup>1</sup>A live system can be accessed via [lstm.seas.harvard.edu](http://lstm.seas.harvard.edu)

such as open parentheses and the start of URLs. [Li et al. \(2016a\)](#) present additional techniques, particularly the use of gradient-based saliency to find important words. [Kádár et al. \(2015, 2017\)](#) show that RNNs specifically learn lexical categories and grammatical functions that carry semantic information, partially by modifying the inputs fed to the model. While inspired by these techniques, our approach tries to extend beyond single examples and provide a general interactive visualization approach of the raw data for exploratory analysis.

There has also been prior work on interactive visualization for interpreting machine learning models. [Tzeng and Ma \(2005\)](#) present a visualization system for feed-forward neural networks with the goal of interpretation, and [Kapoor et al. \(2010\)](#) demonstrate how to tune the training process within a user-interface. The Prospector system ([Krause et al., 2016](#)) provides a general-purpose tool to better understand their machine learning model and its predictions.

Most relevant to this work are systems that focus on analysis of hidden states for convolutional neural networks. [Liu et al. \(2017\)](#) utilize a directed acyclic graph metaphor to show the connections between hidden states and their learned features. [Rauber et al. \(2016\)](#) use low-dimensional projections to explore relationships between neurons and learned observations. Other work has focused on user interfaces for monitoring models, such as TensorBoard ([Abadi et al., 2015](#)) and the related playground for convolutional neural models at [playground.tensorflow.org/](http://playground.tensorflow.org/).

## 5.2 USER ANALYSIS AND GOALS

In the design process for LSTMVis, we decided to focus on the user role of *architects*. We aimed to provide these users with greater visibility into the internals of the system. User feedback from our first prototype further motivated us to include the *trainer* role as users with a focus on the predictions of the model.

With these roles in mind, we aimed to help trainers and architects better understand the high-level

question: “What information does an RNN capture in its hidden feature-states?”. Based on a series of discussions with deep learning experts we identified the following domain goals:

- **G1 - Formulate a hypothesis** about properties that the hidden states might have learned to capture for a specific model. This hypothesis requires an initial understanding of hidden state values over time and a close read of the original input.
- **G2 - Refine the hypothesis** based on insights about learned textual similarities based on patterns in the dynamics of the hidden states. Refining a hypothesis may also mean rejecting it.
- **G3 - Compare models and datasets** to allow early generalization about the insights the representations provide, and to observe how task and domain may alter the patterns in the hidden states.

During the design phase, we developed the following list of tasks that LSTMVis should support from the three domain goals (G1–G3). The mapping of these tasks to goals is indicated by square brackets:

- **T1 - Visualize hidden states** over multiple time steps to allow exploration of the hidden state dynamics in their raw form. [G1]
- **T2 - Filter hidden states** by selecting a subset of timesteps and a threshold of the minimum activation levels throughout the selected steps. These selection methods allow the user to form hypotheses and to separate visual signal from noise. [G1,G2]
- **T3 - Match selections to similar examples** based on hidden state activation pattern. A matched phrase should have intuitively similar characteristics as the selection to support or reject a hypothesis. [G2]
- **T4 - Align textual annotations** visually to matched phrases. These annotations allow the user to compare the learned representation with alternative structural hypotheses such as part-

of-speech tags or known grammars. The set of annotation data should be easily extensible.  
[G2,G3]

- **T5 - Provide a general interface** that can be used with any RNN model and text-like dataset.  
It should make it easy to generate crowd knowledge and trigger discussions on similarities and differences between a wide variety of models. [G3]

This list of tasks provided a guideline for the design of LSTMVis. In addition, tasks (T1-T4) define the core interaction mechanisms for discovery: Visualize (Section 5.3.1) – Filter & Select (Section 5.3.2) – Match & Align (Section 5.3.3). We will first describe the implementation of these interactions and demonstrate their application to multiple use cases in Section 5.4.

### 5.3 DESIGN OF LSTMVIS

LSTMVis, shown in Figure 5.1, is composed of two major visual analysis components. The *Select View* supports the formulation of a hypothesis (T2, G1) by using a novel visual encoding for hidden state changes (T1). The *Match View* (Section 5.3.3) allows refinement of a hypothesis (T3, T4, G2) while remaining agnostic to the underlying data or model (T5).

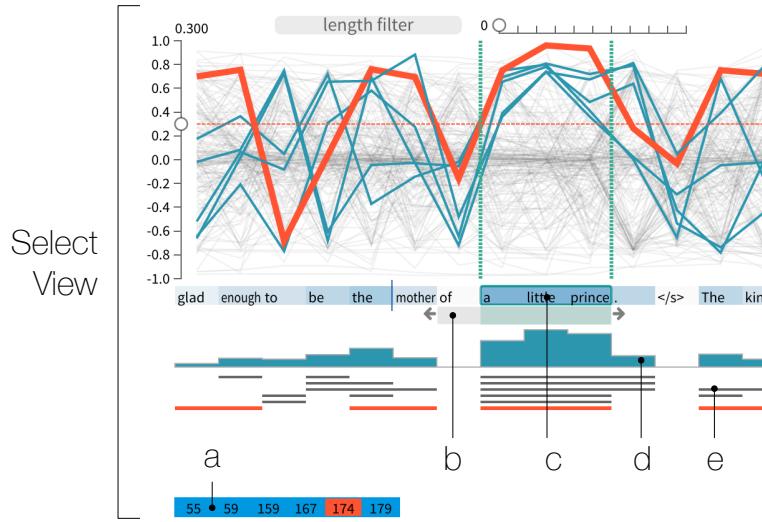
The design of interactive visualization tools is challenging to evaluate. It is crucial to avoid confounding the effects of choices about the interface and those that stem from the possible interactions. The central measure that the design should optimize is whether the tool is useful to the intended users, in our case architects and trainers. We thus followed an iterative design process, in which multiple low-fidelity prototypes were developed and evaluated in collaboration with domain experts, i.e., architects and trainers. Upon their approval of the final design, we released it to a broader audience online to collect long-term feedback. This feedback was used to subsequently improve the system which we describe in the following.



**Figure 5.1:** The LSTMVis user interface. The user interactively selects a range of text specifying a hypothesis about the model in the Select View (a). This range is then used to *match* similar hidden state patterns displayed in the Match View (b). The selection is made by specifying a start-stop range in the text (c) and an activation threshold ( $t$ ) which leads to a selection of hidden states (blue lines). The start-stop range can be further constrained using the pattern plot (d). The meta-tracks below depict extra information per word position like POS (e1) or the top K predictions (e2). The tool can then *match* this selection with similar hidden state patterns in the dataset of varying lengths (f), providing insight into the representations learned by the model. The match view additionally includes user-defined meta-data encoded as heatmaps (g1,g2). The color of one heatmap (g2) can be mapped (h) to the word matrix (f) which allows the user to see patterns that lead to further refinement of the selection hypothesis. Navigation aids provide convenience (i1, i2).

### 5.3.1 VISUALIZATION OF HIDDEN STATE CHANGES

Visualizing the progression of hidden state vectors  $\mathbf{h}_1, \dots, \mathbf{h}_T$  along a sequence of words (time-steps) is at the core of LSTMVis. In the following, we refer to one hidden state as one dimension of the  $D$ -dimensional hidden state vector. To visualize the state dynamic, we decided to consider each hidden state as a data item and time-steps as dimensions for each data item in a parallel coordinates plot. Doing so, we encode the hidden state value using the effective visual variable *position*. The next challenge that the visual design addresses is the formulation of a hypothesis. Allowing the user to perform selection by directly manipulating the hidden state values felt decoupled from the original source of information—the text. The key idea to facilitate this selection process was to allow the user to eas-



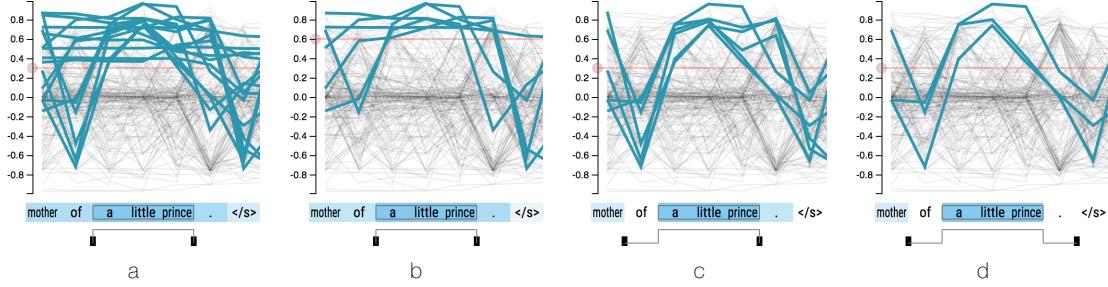
**Figure 5.2:** Snippet of the first complete prototype. While the list of selected cells (a) and the brushing method (c) remained in the final version, we modified the pattern plot (b) and omitted the redundant encodings (d) and (e).

ily discretize the data based on a threshold and select on and off ranges directly on top of the words (Section 5.3.3).

Figure 5.2 shows the first complete prototype that we put online to collect long-term user feedback. Several user comments pointed out that the redundant encoding of hidden states that are “on” in the Select View (Figure 5.2d,e) was not well understood. In the final design shown on the top in Figure 5.1 we omitted this redundant encoding for the sake of clarity and to highlight wider regions of text. The x-axis is labeled with the word inputs  $x_1, \dots, x_T$  for the corresponding time-step. If words do not fit into the fixed width for time steps they are distorted. Figure 5.1 shows the movement of a hidden state vector through an example sequence.

### 5.3.2 SELECT VIEW

The Select View, shown in the top half of Figure 5.1, is centered around the parallel coordinates plot for hidden state dynamics. The full plot comprising all the hidden states can be difficult to compre-



**Figure 5.3:** The hypothesis selection process. (a) The selection covers a `little prince` and has a threshold  $\ell = 0.3$ . Blue highlighted hidden states are selected. (b) The threshold  $\ell$  is raised to 0.6. (c) The pattern plot in the bottom is extended left, eliminating hidden states with values above  $\ell$  after reading of (one word to the left). (d) The pattern plot is additionally extended right, removing hidden states above the threshold after reading “.” (one word to the right). I.e., only hidden states are selected with the following pattern: below threshold  $\ell$  for one word before – above  $\ell$  during a `little prince` – below  $\ell$  for one word after.

hend directly. Therefore, LSTMVis allows the user to formulate a hypothesis (G1) about the semantics of a subset of hidden states by selecting a range of words that may express an interesting property. For instance, the user may select a range of words within tree-structured text (Section 5.4.1), a representative noun phrase in a text corpus (Section 5.4.2), or a chord progression in a musical corpus (Section 5.4.3).

Figure 5.3 demonstrates the selection process. To select, the user brushes over a range of words that form the pattern of interest. In this process, she implicitly selects hidden states that are “on” in the selected range. The dashed red line on the parallel coordinates plot indicates a user-defined threshold value,  $\ell$ , that partitions the hidden states into “on” ( $\text{timesteps} \geq \ell$ ) and “off” ( $\text{timesteps} < \ell$ ) within this range. In addition to selecting a range, the user can modify the selection to define that hidden states must also be “off” immediately before or after the selected range. This is necessary to differentiate those hidden states that are active for the pattern under investigation from those cells that are active for more time steps. Figure 5.3 shows different range selections and the corresponding hidden states. Using these selection criteria the user creates a set of selected hidden states  $\mathcal{S}_1 \subset \{1 \dots D\}$  that follow the specified on/off pattern with regard to the defined threshold.

A user can also add aligned tracks of textual annotations (meta tracks) to the selection view. For example, she can visualize part-of-speech (POS) annotations or named entity recognition (NER) results. The feature of meta tracks is the result of feedback from multiple online users asking us to include the tracks. Some users that used the tool for training also wanted to see the top K predictions (outcomes) for each word. Figure 5.1 shows examples for POS (e1) and top K (e2).

These interaction methods allow the user to define a hypothesis as word range which results in the selection of a subset of hidden states following a specified pattern above a defined threshold (T2, G1) that only relies on the hidden state vectors themselves (T5). This hypothesis can be further informed by the meta tracks. To refine or reject the hypothesis the user can then make use of the Match View.

### 5.3.3 MATCH VIEW

The Match View (Figure 5.1b) provides evidence for or against the selected hypothesis. The view presents a set of relevant matched phrases that have similar hidden state patterns as the phrase selected by the user. These phrases are searched for within a dataset that is representative of the data that the model was trained on. This ensures that the patterns in the dataset correspond to those learned by the model.

With the goal of maintaining an intuitive match interface, we define “matches” to be ranges in the dataset that would have lead to a similar set of hidden states under the selection criteria (threshold, on/off pattern). Formally, we assume that the user has selected a threshold  $\ell$  with hidden states  $S_1$  and has not limited the selection to the right or left further, as shown in Figures 5.3a and 5.3b. We rank all possible candidate ranges in the dataset starting at time  $a$  and ending at time  $b$  with a two step process

1. Collect the set of all hidden states that are “on” for the range,

$$S_2 = \{c \in \{1 \dots D\} : h_{t,c} \geq \ell \text{ for all } a \leq t \leq b\}$$

- Rank the candidates by the number of overlapping states  $|\mathcal{S}_1 \cap \mathcal{S}_2|$  using the inverse of the number of additional “on” cells  $-|\mathcal{S}_1 \cup \mathcal{S}_2|$  and candidate length  $b - a$  as tiebreaks.

If the original selection is limited on either side (as, for example, in Figures 5.3c or 5.3d), we modify step (2) to take this into account for the candidates. For instance, if there is a limit on the left, we only include state indices  $c$  in  $\mathcal{S}_2$  in that also satisfy  $h_{a-1,c} < \ell$ . For efficiency, we do not score all possible candidate ranges (datasets typically have  $T > 1$  million). We limit the candidate set by filtering to ranges with a minimum number of hidden states from  $\mathcal{S}_1$  over the threshold  $\ell$ . These candidate sets can be computed efficiently using run-length encoding.

From the matching algorithm, we retrieve the top 50 results which are shown, one per row each, in a word matrix (e.g., Figure 5.1f) located in the Match View. Each cell in the word matrix is linked to a cell in corresponding heatmaps. These heatmaps encode additional information about each word in the matching results. The always available *match count* heatmap (Figure 5.1-g1) encodes the number of overlapping states  $|\mathcal{S}_1 \cap \mathcal{S}_2|$  for each timestep.

The user can use additional annotations, similar to meta tracks in the Selection View, as heatmaps (T4). These annotations act as ground truth data, e.g., part-of-speech tags for a text corpora (Figure 5.1-g2), or as further information to help calibrate the hypotheses, e.g., nesting depth of tree-structured text. Figure 5.1 shows how hovering over “little” (mouse pointer) leads to highlights in the *match count* heatmap (g1) indicating seven overlapping states between match and selection for this position. The POS heatmap (g2) indicates that the word at this position is as adjective (ADJ).

The heatmap colors can be mapped directly to the background of the matches (Figure 5.1h) as a simple method to reveal pattern across results (Figure 5.1f). Based on human identifiable patterns or alignments, the matching and mapping methods can lead to further data analysis or a refinement of the current hypothesis.

## 5.4 USE CASES

In this section we highlight three use cases that demonstrate the general applicability of LSTMVis for the analysis of hidden states.

### 5.4.1 PROOF-OF-CONCEPT: PARENTHESIS LANGUAGE

As proof of concept we trained an LSTM as language model on synthetic data generated from a very simple counting language with a parenthesis and letter alphabet  $\Sigma = \{(\ )\ 0\ 1\ 2\ 3\ 4\}$ . The language is constrained to match parentheses, and nesting is limited to be at most 4 levels deep. Each opening parenthesis increases and each closing parenthesis decreases the nesting level, respectively. Numbers are generated randomly, but are constrained to indicate the nesting level at their position.

For example, a string in the language might look like:

( 1 ( 2 ) () ) 0 ( ( ( 3 ) ) 1 )

The string consists of nested parentheses and numbers. Three blue horizontal lines are placed under the first, second, and fourth opening parentheses, and under the corresponding closing parentheses. Two orange horizontal lines are placed under the third and fifth opening parentheses, and under the corresponding closing parentheses. One green horizontal line is placed under the sixth opening parenthesis and its corresponding closing parenthesis.

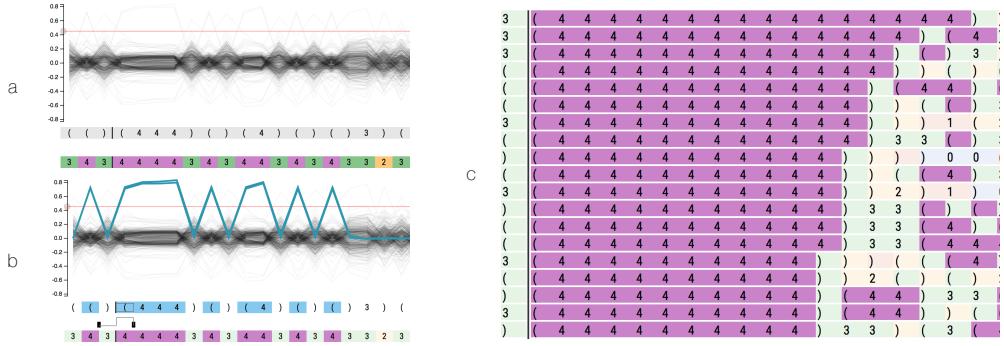
Blue lines indicates ranges of nesting level  $\geq 1$ , orange lines indicate nesting level  $\geq 2$ , and green lines indicate nesting level  $\geq 3$ .

To analyze this language, we view the states in LSTMVis. An example of the cell states of a two-layer LSTM model with 300 hidden states is shown in Figure 5.4(a). In this example, even the initial parallel coordinates plot shows a strong regularity, as hidden state changes occur predominately at parentheses<sup>2</sup>.

Given this observation, we form the hypothesis that some hidden states are dedicated to track a specific nesting level. To test this, we select a range spanning nesting level four by selecting the phrase ( 4. We immediately see that several hidden states seem to cover this pattern and that in the local

---

<sup>2</sup>In layer work, we demonstrated that even very small LSTMs are able to count the nesting level for much deeper nested languages since they are able to use the hidden states to dynamically count (Suzgun et al., 2019)



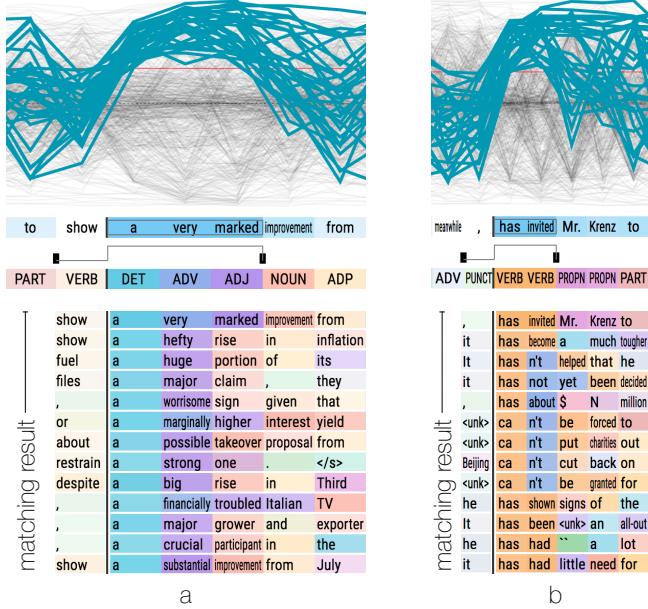
**Figure 5.4:** Plot of a phrase from the parenthesis synthetic language. (a) The full set of hidden states. Note the strong movement of states at parenthesis boundaries. (b) A selection is made at the start of the fourth level of nesting. Even in the select view it is clear that several hidden states represent a four-level nesting count. In both plots the meta-track indicates the nesting level as ground truth. (c) The result of matching the selected states indicates that they seem to capture nesting level 4 phrases of variable length.

neighborhood several other occurrences of our hypothesis are covered as well, e.g., the empty parenthesis and the full sequence `( 4 4 4 )`. This observation confirms earlier work that demonstrates that LSTMs can learn simple context-free grammars (Wiles, 1998, Gers and Schmidhuber, 2001).

#### 5.4.2 PHRASE SEPARATION IN LANGUAGE MODELING

Next we consider the case of a real-world natural language model from the perspective of an architect interested in the structure of the internal states and how they relate to underlying properties. For this experiment we trained a 2-layer LSTM language model with 650 hidden states on the Penn Treebank (Marcus et al., 1993) following the setup of (Zaremba et al., 2014). While the model is trained for language modeling, we were interested in seeing if it additionally learned properties about the underlying language structure. To test this, we additionally include linguistic annotations in the visual analysis from the Penn Treebank. We experimented with including part-of-speech tags, named entities, and parse structure.

Here we focus on the case of *phrase chunking*. We annotated the dataset with the gold-standard phrase chunks provided by the CoNLL 2003 shared task (Tjong Kim Sang and De Meulder, 2003)

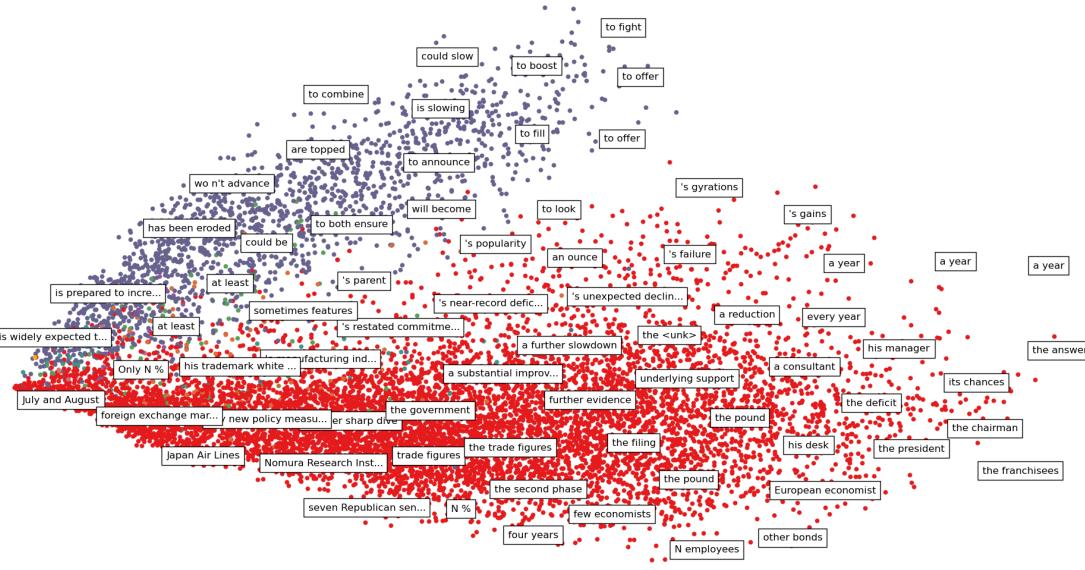


**Figure 5.5:** Phrase selections and match annotations in the Wall Street Journal. (a) The user selects the phrase `a very marked improvement` (turning off after `improvement`). The matches found are entirely other noun phrases, and start with different words. Ground-truth noun phrases are indicated with a sequence of colors: cyan (DET), blue (ADV), violet (ADJ), red (NOUN). (b) We select a range starting with `has invited`. The results are various open verb phrases as sequence of colors orange (VERB) and blue (ADV). Note that for both examples the model can return matches of varying lengths.

for a subset of the treebank (Sections 15-18). These include annotations for noun phrases and verb phrases, along with prepositions and several other less common phrase types.

While running experimental analysis, we found a strong pattern that selecting noun phrases as hypotheses leads to almost entirely noun phrase matches. Additionally, we found that selecting verb phrase prefixes would lead to primarily verb phrase matches. In Figure 5.5 we show two examples of these selections and matches.

The visualization hints that the model has implicitly learned a representation for language modeling that can differentiate between the two types of phrases. While the tool itself cannot confirm or deny this type of hypothesis, the aim is to provide clues for further analysis. We can check, outside of the tool, if the model is clearly differentiating between the classes in the phrase dataset. To do this we compute



**Figure 5.6:** PCA projection of the hidden state patterns ( $\mathcal{S}_1$ ) of all multi-word phrasal chunks in the Penn Treebank, as numerical follow-up to the phrase chunking hypothesis. Red points indicate noun phrases, blue points indicate verb phrases, other colors indicate remaining phrase types. While trained for language modeling, the model separates out these two phrase classes in its hidden states.

the set  $\mathcal{S}_1$  for every noun and verb phrase in the dataset. We then use PCA on the vector representation to generate a two-dimensional embedding for each set. The results are shown in Figure 5.6, which shows that these on-off patterns are sufficient to partition the noun and verb phrases<sup>3</sup>.

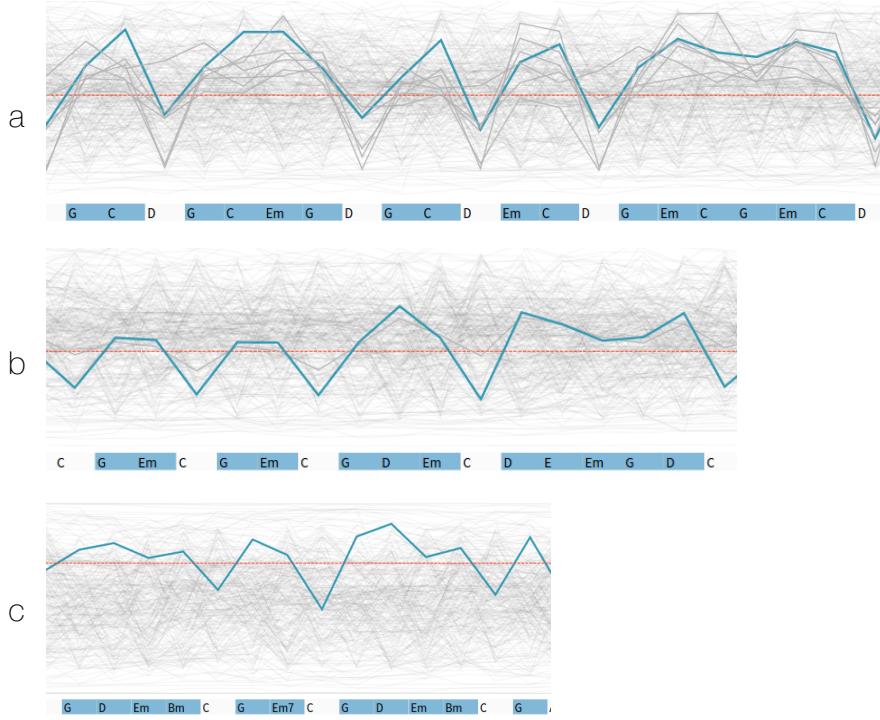
#### 5.4.3 MUSICAL CHORD PROGRESSIONS

Past work on LSTM structure has emphasized cases where single hidden states are semantically interpretable. We found that with a few exceptions (quotes, brackets, and commas) this was rarely the case. However, for datasets with more regular long-term structure, single states could be quite meaningful.

As a simple example, we collected a large set of songs with annotated chords for rock and pop songs

---

<sup>3</sup>Work by Belinkov et al. (2017b) first showed that even neural machine translation systems successfully learn syntactic information.



**Figure 5.7:** Three examples of single state patterns in the guitar chord dataset. (a) We see several permutations of the very common I - V - vi - IV progression (informally, the “Don’t Stop Believing” progression). (b) We see several patterns ending in a variant of the I - vi - IV - V (the 50’s progression). (c) We see two variants of I - V - vi - iii - IV - I (beginning of the Pachelbel’s Canon progression). Chord progression patterns are based on <http://openmusictheory.com/>.

to use as a training dataset, 219k chords in total. We then trained an LSTM language model to predict the next chord  $x_{t+1}$  in the sequence, conditioned on previous chord symbols (chords are left in their raw format).

When we viewed the results in LSTMVis we found that the regular repeating structure of the chord progressions is strongly reflected in the hidden states. Certain states will turn on at the beginning of a standard progression, and remain on through variant-length patterns until a resolution is reached. In Figure 5.7, we examine three very common general chord progressions in rock and pop music. We select a prototypical instance of the progression and show a single state that captures the pattern by remaining on when the progression begins and turning off upon resolution.

## 5.5 LONG-TERM CASE STUDY

Shneiderman and Plaisant (2006) propose strategies for multi-dimensional in-depth long-term case (MILC) studies to evaluate information visualization tools. They observe that “scaling up by an order of magnitude in terms of number of users and duration of the observations is clearly beneficial.” We decided to adopt their core ideas and report on qualitative feedback and quantitative success indicators after the open-source release of LSTMVis in June 2016.

We created a webpage and a video that introduces the core ideas of LSTMVis at [lstm.seas.harvard.edu](http://lstm.seas.harvard.edu). The webpage provides an overview video, links to the code, the online demo, and an opportunity for users to leave comments. To advertise the tool online we followed a social media strategy that included collecting followers on Twitter, using broadcast media such as Reddit and Hackernews, and inviting editors of popular machine learning blogs to write guest articles.

The webpage contains a demo system with example datasets that allows us to acquire a large set of logging data. We noticed that users often try to reproduce the scenarios that are explained in the online video. To allow users to share insights from their exploratory analysis, we ensured that our URL parameter string contains all necessary information to replay and share a scenario.

We collected logging information about our webpage using Google Analytics. Within the first 7 days, the web page received  $\sim$ 5,600 unique users, with 49% of traffic coming through social media and 39% arriving directly. The social media traffic was dominated by channels that we used to advertise the tool (Twitter 40%, Reddit 26%). To our surprise we also observed substantial traffic from channels that we did not contact directly (e.g., Sina Weibo 11%, Google+ 9%). After 300 days we recorded  $\sim$ 19,500 page views with shrinking user traffic from social media (31%). At that point most users used search (22%) or accessed the webpage directly (39%). Only a small percentage (10%) of users tried the online demo. Most of these users used the datasets shown in the explanation video and did not explore further.

Our open source code release was stable yet simple enough to “...ensure that the tool has a reasonable level of reliability and support for basic features” (Shneiderman and Plaisant, 2006). For easy adoption, we provide a detailed documentation and convenience tools to prepare the data for import. We asked students to act as testers for our source code. Based on their feedback we made several improvements to the installation process. For example, our prototype required NodeJS to resolve client-side dependencies. By providing the required libraries within our repository we removed the cumbersome step of installing NodeJS for our target audience. We observe that around 850 programmers liked the project (stars on GitHub) and over 200 practitioners copied (forked) the project to make custom modifications.

Furthermore, we observe adoption of the tool for several documented use cases. Evermann et.al. (Evermann et al., 2017) describe the application of LSTMVis to understand a trained model for a business process intelligence dataset. Liu et.al. (Liu et al., 2016) use LSTMVis in experiments to investigate language variants and vagueness in website privacy policies. We see an increasing interest to apply our tool for biomedical and genomic data (Ching et al., 2018).

Besides the quantitative observations we also collected qualitative feedback from comments on the webpage, GitHub tickets (feature requests), and in-person presentations of the prototype. This qualitative feedback led us to make several changes to our system that were discussed in Section 5.3.

In retrospective, conducting a long-term user study benefited the project at multiple stages. Preparing the release of the prototype required us to focus strongly on simplicity, usability, and robustness of our tool. This lead to many small improvements to an internal prototype. The design iterations we inferred from user feedback strengthen the tool further. We think, that planning a successful long-term study requires four core ingredients: (1) reach out to your target audience by describing your approach (webpage, video) and inform them using social media, email, etc. (2) allow users to play with your tool by setting up a demo server, (3) allow engagement and experimentation with your tool by providing sufficiently documented, easily adoptable source code, and (4) make it as simple as pos-

sible for users to give you feedback via discussion forums, reported issues, or in person. During the study, we found it to be crucial to continuously engage with users and quickly take action based on their feedback.

## 5.6 CONCLUSION

LSTMVis provides an interactive visualization to facilitate data analysis of RNN hidden states. The tool is based on direct inference where a user *selects* a range of text to represent a hypothesis and the tool then *matches* this selection to other examples in the dataset. The tool easily allows for external annotations to verify or reject hypothesized. It only requires a time-series of hidden states, which makes it easy to adopt for a wide range of visual analyses of different datasets and models, and even different tasks (language modeling, translation etc.).

To demonstrate the use of the model we presented several case studies of applying the tool to different datasets. Releasing the tool and source code online allows us to collect long-term user feedback that has already led us to make several improvements. In future work, we will explore how the wide variety of application cases can be adopted – beyond our imagined use cases and user groups. As example for such a use case, we got contacted by a highschool student using LSTMVis to learn about RNN methods.

Another lesson taken from the development of the tool was the value of developing an intuition for machine learning model behavior. In contrast to the phenotyping example from chapter 4, LSTMVis allows users to interact with the internal states of a model. That means not only that the tool allows the testing of multiple hypotheses, it also helps users form these hypotheses by forming an intuition about the model behavior.

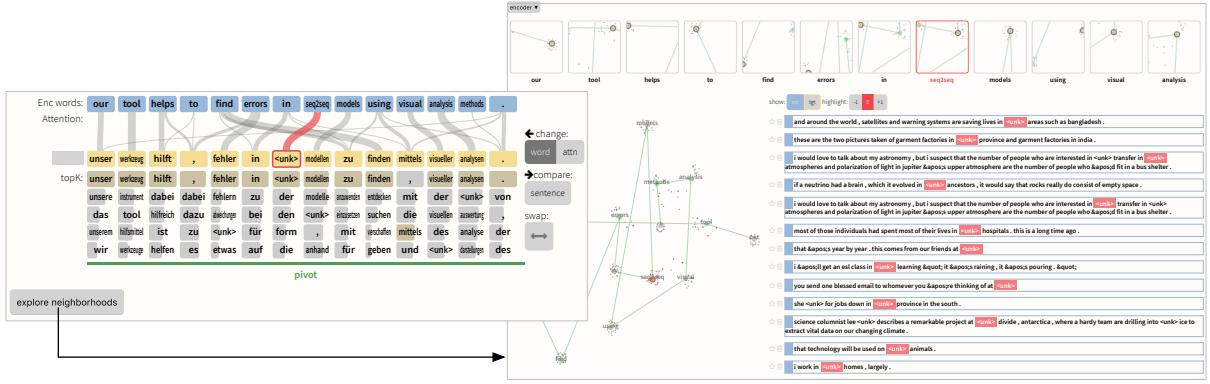
A limitation of LSTMVis is that it tries to explain a model using globally learned rules. It does not, however, explain predictions. Focusing on the same user type of architects and trainers, we will

present a tool that assists in the understanding of predictions in Chapter 6.

# 6

## Debugging Predictions of Sequence-to-Sequence Models

So far, we have presented two examples of methods aiming to understand what a model has learned as a whole. Building this intuition is crucial to validate whether the the model performs as expected, it does not help when trying to debug model decisions, particularly when the model makes mistakes. This split between trying to understand the model or its prediction is commonly seen in the interpretability



**Figure 6.1:** Example of Seq2Seq-Vis. In the translation view (left), the source sequence “our tool helps to find errors in seq2seq models using visual analysis methods.” is translated into a German sentence. The word “seq2seq” has correct attention between encoder and decoder (red highlight) but is not part of the language dictionary. When investigating the encoder neighborhoods (right), the user sees that “seq2seq” is close to other unknown words “<unk>”. The buttons enable user interactions for deeper analysis.

literature, as we described in Chapter 3.2. However, while the aim differs, we can apply the same insight that interaction builds intuition to this problem.

In this work we focus on attention-based *sequence-to-sequence models* (seq2seq, Sutskever et al., 2014, Bahdanau et al., 2015). Seq2seq models have shown state-of-the-art performance in a broad range of generation tasks, for example in machine translation, image captioning, and summarization. Recent results show that these models exhibit human-level performance in machine translation for certain important domains (Wu et al., 2016, Hassan Awadalla et al., 2018). Seq2seq models are powerful because they provide an effective supervised approach for processing and predicting sequences without requiring manual specification of the relationships between source and target sequences. Using a single model, these systems learn to do reordering, transformation, compression, or expansion of a source sequence to an output target sequence.

However, similar to the other models we have investigated so far, Seq2seq models act as black-boxes during prediction, making it difficult to track the source of mistakes. The high-dimensional internal representations make it difficult to analyze the model as it transforms the data. While this property

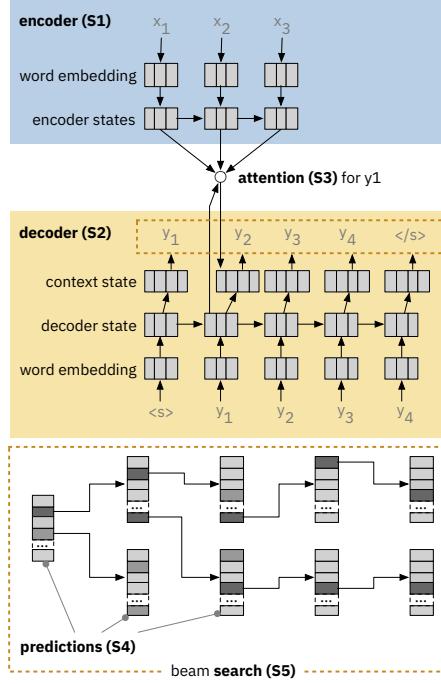
is shared across deep learning, mistakes involving language are often very apparent to human readers. For instance, a widely publicized incident resulted from a seq2seq translation system mistakenly translating “good morning” into “attack them” leading to a wrongful arrest (Hern, 2017). Common but worrying failures in seq2seq models include machine translation systems greatly mistranslating a sentence, image captioning systems yielding an incorrect caption, or speech recognition systems producing an incorrect transcript.

Ideally, model developers would understand and trust the results of their systems, but currently, this goal is out of reach. We, therefore, contribute to the crucial challenge of better surfacing the mistakes of seq2seq systems in a general and reproducible way. We propose SEQ2SEQ-VIS, a visual analytics tool provides support for the following three goals:

- **Examine Model Decisions:** allow users to understand, describe, and externalize model errors for each stage of the seq2seq pipeline.
- **Connect Decisions to Samples:** describe the origin of a seq2seq model’s decisions by relating internal states to relevant training samples.
- **Test Alternative Decisions:** facilitate model interventions by making it easy to manipulate of model internals and conduct ”what if” explorations.

The full system is shown in Figure 6.1 (or larger in Figure 6.7). It integrates visualizations for the components of the model (Fig 6.1 left) with internal representations from specific examples (Fig 6.1 middle) and nearest-neighbor lookups over a large corpus of precomputed examples (Fig 6.1 right). The source code, a demo instance, and a descriptive webpage are available at <http://seq2seq-vis.io>

Throughout this work, we consider the running example of automatic translation of one sequence  $\mathbf{x}_{1:S}$  in one language to a sequence  $\mathbf{y}_{1:T}$  in another language. Recall the stages of seq2seq models from Chapter 2.3.5, also shown in Figure 6.2: An encoder (S1) encodes the word embedding  $\mathbf{x}_s$  into a contextualized hidden representation  $\mathbf{h}_s^{(S)}$ . A decoder (S2) generates the representations  $\mathbf{h}_{1:t}^{(T)}$  for the



**Figure 6.2:** Five stages in translating a source to target sequence: (S1) encoding the source sequence into latent vectors, (S2) decoding to generate target latent vectors, (S3) attend between encoder and decoder, (S4) predict word probabilities at each time step, and (S5) search for a best complete translation (beam search).

sequence of generated target words  $\mathbf{y}_{1:t}$ . The attention (S3) computes an alignment between a target step and all source tokens. We refer to the attention weight from decoding step  $t$  to a source word  $x_s$  as  $a_{t,s}$ . The prediction step (S4) computes a distribution  $p(\mathbf{y}_{t+1}|\mathbf{x}_{1:S}, \mathbf{y}_{1:t})$  over the vocabulary of the target language. Finally, the beam search (S5) is used to approximate the optimal sequence of tokens for a translation by maintaining a fringe of the top  $K$  hypotheses until all have terminated by generating the stop token.

Each stage of the process is crucial for effective translation, and it is hard to separate them. However, the model does preserve some separations of concerns. The decoder (S2) and encoder (S1) primarily work with their respective language, and manage the change in hidden representations over time. Attention (S3) provides a link between the two representations and connects them during training. Pre-

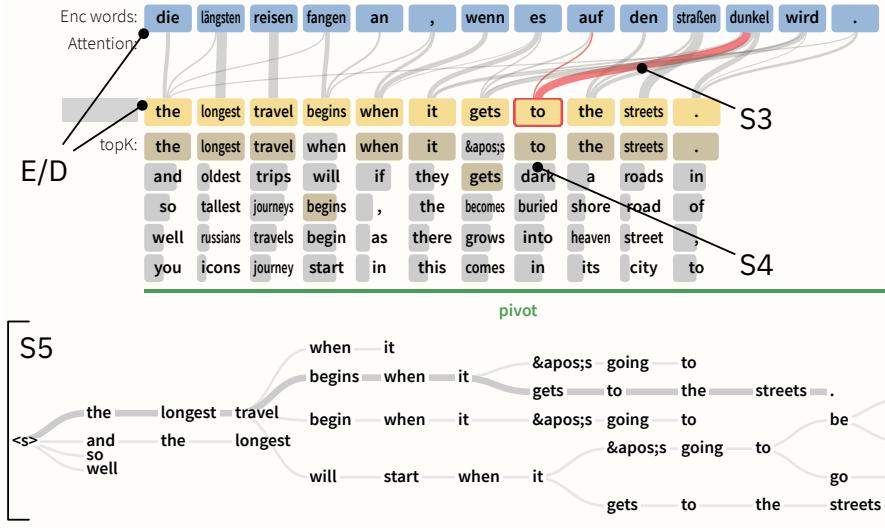


Figure 6.3: An overview of the Seq2seq-Vis interface for the case study.

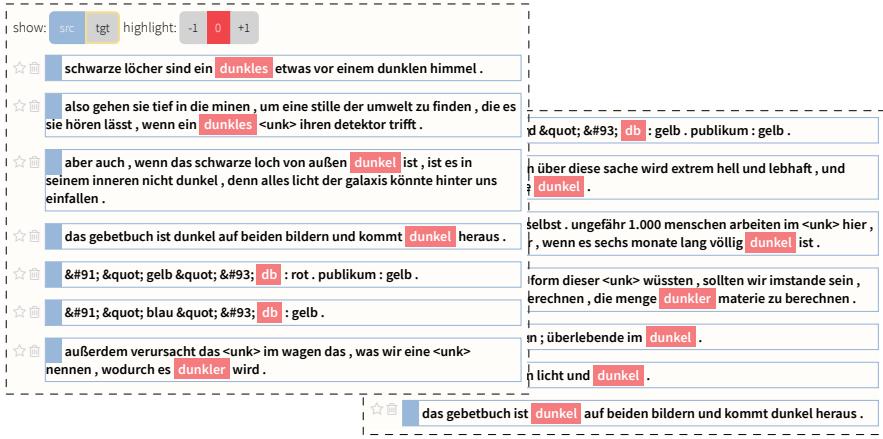
diction (S4) combines the current decoder state with the information moving through the attention. Finally, search (S5) combines these with a global score table. These five stages provide the foundation for our visual analytics system.

## 6.1 MOTIVATING CASE STUDY: DEBUGGING TRANSLATION

To motivate the need for a neural debugger, we present a representative case study. Further case studies are discussed in section 6.4. This case study involves a model trainer (see Section 3.1) named Beth who is building a German-to-English translation model. The model is trained on the small-sized IWSLT '14 dataset (Mauro et al., 2012) with 200,000 examples and thus makes frequent mistakes.

Beth observes that a specific example was mistranslated in a production setting. She finds the source sentence: *Die längsten Reisen fangen an, wenn es auf den Straßen dunkel wird.*<sup>1</sup> This sentence should have been translated to: *The longest journeys begin, when it gets dark in the streets.* She notices that

<sup>1</sup>The closing quote of the book ‘Kant’ from German author Jörg Fauser, who is attributed as being a fore-runner of German underground literature.



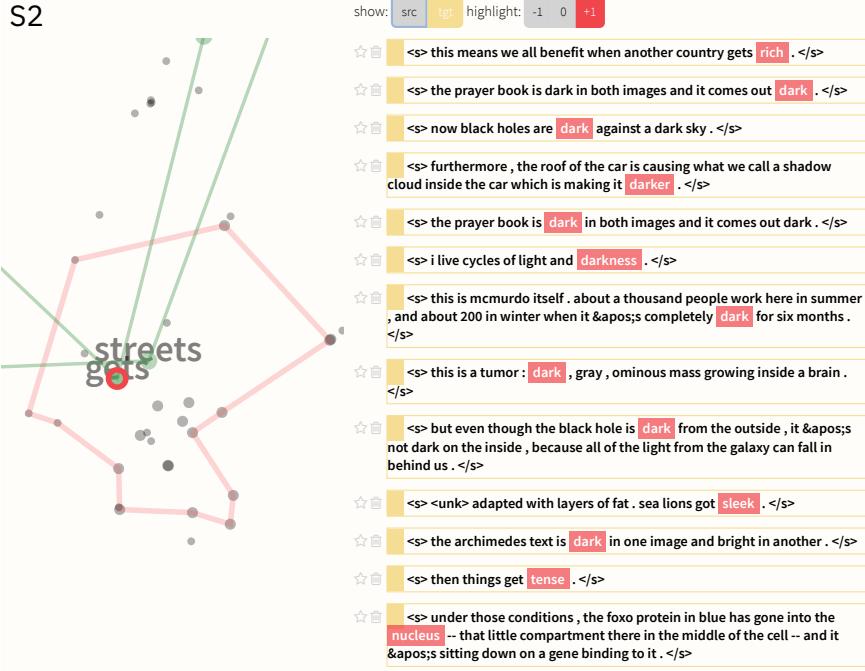
**Figure 6.4:** Hypothesis: Encoder (S1) Error – nearest neighbors of encoder state for *dunkel*.

the model produces the mistranslation: *The longest journey begins, when it gets to the streets*. Figure 6.3(E/D) shows the tokenized input sentence in blue and the corresponding translation of the model in yellow (on the top). The user observes that the model does not translate the word *dunkel* into *dark*.

This mistake exemplifies several goals that motivated the development of Seq2Seq-Vis. The user would like to examine the system’s decisions, connect to training examples, and test possible changes. These goals apply to all five model stages: encoder, decoder, attention, prediction, and search.

**Hypothesis: Encoder (S1) Error?** Following the same approach as with LSTMVis, Seq2Seq-Vis lets the user examine examples with similar encoder states. Throughout, we will use the term *neighborhood* to refer to the twenty closest states in vector space from training data. Seq2Seq-Vis displays the nearest neighbor sentences for a specific encoder state as red highlights in a list of training set examples. Figure 6.4 shows that the nearest neighbors for *dunkel* match similar uses of the word. The majority seem to express variations of *dunkel*. The few exceptions, e.g., *db*, are artifacts that can motivate corrections of the training data or trigger further investigation. Overall, the encoder seems to perform well in this case.

**Hypothesis: Decoder (S2) Error?** Similarly, Beth can apply Seq2Seq-Vis to investigate the neigh-



**Figure 6.5:** Hypothesis: Decoder (S2) Error – nearest neighbors of decoder state for *gets* and *streets*, which are close in projection space.

borhood of decoder states produced at times  $t$  and  $t + 1$  (Figure 6.5). In addition to the neighbor list, it gives a projection view that depicts all decoder states for the current translation and all their neighbors in a 2D plane. The analyst observes that the decoder states produced by *gets* and *streets* are in proximity and share neighbors. Since these states are indicative for the next word we can switch the highlight one text position to the right (by clicking the +1 button) and observe that the decoder states at *gets* and *streets* support producing *dark*, *darker*, or *darkness*. Thus, the decoder state does not seem very likely as the cause of the error.

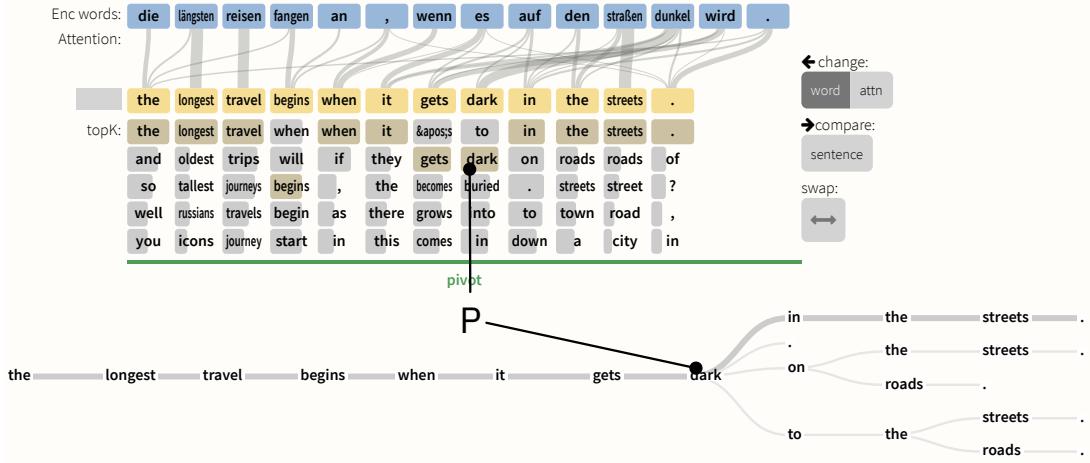
**Hypothesis: Attention (S3) Error?** Since both encoder and decoder are working as expected, another possible issue is that the attention may not focus on the corresponding source token *dunkel*. The previous test revealed that well-supported positions for adding *dark* are after *gets* or *streets*. This matches human intuition, as we can imagine the following sentences which are valid translations: *The*

*longest travels begin when it gets dark in the streets.* or *The longest travels begin when it gets to the streets turning dark.* In Figure 6.3(S3) Beth can observe that the highlighted connection following *get* to the correct next word *dunkel* is very strong. The connection width indicates that the attention weight is very high with the correct word. Therefore, Beth assumes that the attention is well set for predicting *dark* in this position. The hypothesis for an error in S3 can be rejected.

**Hypothesis: Prediction (S4) Error?** The combination of decoder state and attention is used to compute the probability of the next word. It may be that an error occurs in this decision, leading to a poor probability of the word *dark*. The tool shows the most likely next words and their probabilities in Figure 6.3(S4). Here, our analyst can see that the model mistakenly assigns a higher probability to *to* than *dark*. However, both options are very close in probability, indicating that the model is quite uncertain and almost equally split between the two choices. These local mistakes should be automatically fixed by the beam search, because the correct choice *dark* leads to a globally more likely sentence.

**Hypothesis: Search (S5) Error?** Having eliminated all other possible issues, the problem is likely to be a search error. Beth can investigate the entire beam search tree in Figure 6.3(S5), which shows the top  $K$  considered options at each prediction step. In this case, the analyst finds that *dark* is never considered within the search. Since the previous test showed that *to* is only minimally more likely than *dark*, a larger  $K$  would probably have lead to the model considering *dark* as the next best option. We therefore conclude that this local bottleneck of a too narrow beam search is the most likely error case. The analyst has identified a search error, where the approximations made by beam search cut off the better global option in favor of a worse local choice.

**Exploring Solutions.** When observing the  $K$ -best predictions for the position of *to*, Beth sees that *dark* and *to* are close in probability (Figure 6.3(S4)). To investigate whether the model would produce the correct answer if it had considered *dark*, Seq2Seq-Vis allows Beth to evaluate a case-specific fix. The analyst can test this counterfactual, what would have happened if she had forced the translation to use



**Figure 6.6:** Testing a fix – by clicking on the correct word *dark* in the predicted top-K, the beam search is forced on a specific path (P) which leads to the correct prediction.

*dark* at this critical position? By clicking on *dark* she can produce this probe (shown in Figure 6.6), which yields the correct translation. The user can now describe the most likely cause of error (search error) and a local fix to the problem (forced search to include *dark*). Beth can now add this case to a list of well-described bugs for the model and later consider a global permanent fix.

## 6.2 GOALS AND TASKS

We now step back from this specific instance and consider a common deployment cycle for a deep learning model such as seq2seq. First, a model is trained on a task with a possibly new data set, and then evaluated with a standard metric. The model performs well in aggregate and the stakeholders decide to deploy it. However, for a certain subset of examples there exist non-trivial failures. These may be noticed by users, or, in the case of translation, by post-editors who correct the output of the system. While the model itself is still useful, these examples might be significantly problematic as to cause alarm.

Although these failures can occur in any system, this issue was much less problematic in previous

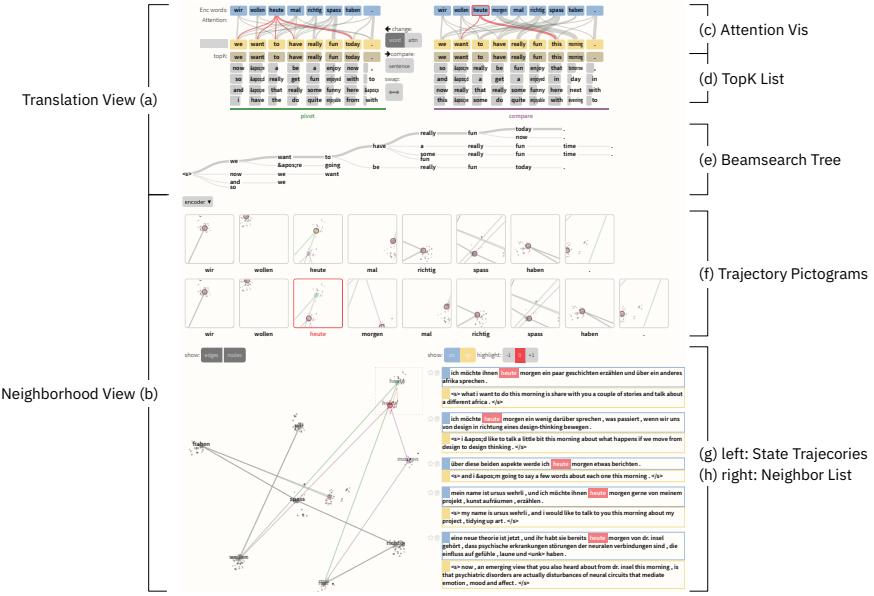
generations of AI systems. For instance when using rule-based techniques, a user can explore the provenance of a decision through rules activated for a given output. If there is a mistake in the system, an analyst can 1) identify which rule misfired, 2) see which previous examples motivated the inclusion of the rule, and 3) experiment with alternative instances to confirm this behavior.

Ideally, a system could provide both functionalities: the high performance of deep learning with the ability to interactively spot issues and explore alternatives. However, the current architecture of most neural networks makes it more challenging to examine decisions of the model and locate problematic cases. Our work tackles the following challenges and domain goals for seq2seq models analogous to the three steps in rule-based systems:

**Goal G1 – Examine Model Decisions:** It is first important to examine the model’s decision chain in order to track down the error’s root cause. Seq2seq models make decision through several stages. While it has proven difficult to provide robust examination in general-purpose neural networks, there has been success for specific decision components. For example, the attention stage (S3) has proven specifically useful for inspection (Xu et al., 2015, Bahdanau et al., 2015, Lee et al., 2017). Our first goal is to develop interactive visual interfaces that help users understand the model’s components, their relationships, and pinpoint sources of error.

**Goal G2 – Connect Decisions to Samples from Training Data:** Once a model makes a particular decision, a user should be able to trace what factors influenced this decision. While it is difficult to provide specific reasoning about the many factors that led to a decision in a trained model, we hope to provide other means of analysis. In particular, we consider the approach of mapping example states to those from previous runs of the model. For instance, the training data defines the world view of a model and therefore influences its learned decisions (Koh and Liang, 2017). The goal is to utilize (past) samples from training data as a proxy to better understand the decision made on the example in question.

**Goal G3 – Test Alternative Decisions:** Ultimately, though, the goal of the user is to improve the



**Figure 6.7:** Overview of Seq2Seq-Vis: The two main views (a) Translation View and (b) Neighborhood View facilitate different modes of analysis. Translation View provides (c) visualizations for attention, (d) the top k word predictions for each time step, and (e) the beam search tree. The Neighborhood View goes deeper into what the model has learned by providing (f,g) a projection of state trajectories and (h) a list of nearest neighbors for a specific model state.

model's performance and robustness. While the current state-of-the art for diagnosing and improving deep neural network models is still in an early stage (Karpfathy et al., 2015, Smilkov et al., 2017a, Li et al., 2016b, Kahng et al., 2018), our goal is to allow the user to test specific interventions. We aim to let the user investigate causal effects of changing parts of the model the let users ask *what if* specific intermittent outputs of a model changed.

Our motivating case study (section 6.1) follows these goals: First, the user defines five hypotheses for causes of error and tests them by examining the model's decisions (G1). Some of these decisions (for S1, S2) are represented in the model only as latent high-dimensional vectors. To make these parts tangible for the user, she connects them to representative neighbors from the training data (G2). Finally, by probing alternatives in the beam search (G3) she finds a temporary alternative that helps her to formulate a better solution.

We use these goals to compile a set of visualization and interaction tasks for SEQ2SEQ-VIS. The mapping of these tasks to goals is indicated by square brackets:

**Task T1 - Create common visual encodings** of all five model stages to allow a user to examine the learned connections between these modules. [G1]

**Task T2 - Visualize state progression** of latent vector sequences over time to allow for high-level view of the learned representations. [G1]

**Task T3 - Explore generated latent vectors and their nearest neighbors** by querying a large database of training examples to facilitate error identification and training adjustment. [G2]

**Task T4 - Generate sensible alternative decisions for different stages of the model and compare them** to ease model exploration and compare possible corrections. [G1, G3]

**Task T5 - Create a general and coherent interface** to utilize a similar front-end for many sequence-to-sequence problems such as translation, summary, and generation. [G1,G2,G3]

In the following section, we will match these tasks and goals to design decisions for Seq2Seq-VIS.

### 6.3 DESIGN OF SEQ2SEQ-VIS

The design process of SEQ2SEQ-VIS applied the lessons learned during the development of LSTMVis. We followed an iterative design process with frequent discussions between experts in machine learning and visualization. In regular meetings we evaluated a series of low-fidelity prototypes and tested them for usability. The design presented in this section combines the prevailing ideas into a comprehensive tool.

The tool also follows the same strategy of *select* and *match*. This strategy is embodied in the two main views that facilitate those modes: In the upper part, the *translation view* provides a visual encoding for each of the model stages and fosters understanding and comparison tasks. In the lower part, the *neighborhood view* enables deep analysis based on neighborhoods of training data. Figure 6.7

shows the complete tool.

**TRANSLATION VIEW** In the translation view (Figure 6.7a), each functional stage of the seq2seq model is mapped to a visual encoding ( $T_1, T_2, G_1$ ). These encodings generalize the strategies that were taken by Olah and Carter (2016) and Le et al. (2012). In Attention Vis (Figure 6.7c), the encoder words are shown in blue, the decoder words in yellow, and the attention is shown through weighted bipartite connections. To reduce visual clutter the attention graph is pruned. For each decoder step all edges are excluded that fall into the lower quartile of the attention probability distribution.

Right below the yellow decoder words, the top  $K$  predictions (S4 of model) for each time step are shown (Figure 6.7d). Each possible prediction encodes information about its probability in the underlying bar chart, as well as an indication if it was chosen for the final output (yellow highlight).

In the bottom part of the translation view, a tree visualization shows the hypotheses from the beam search stage (Figure 6.7e). The most probable hypothesis, which results in the final translation sentence, is highlighted. Several interactions can be triggered from the translation view.

**NEIGHBORHOOD VIEW** The neighborhood view (Figure 6.7b) connects at model decisions to similar examples ( $T_2, T_3, G_1, G_2$ ). Seq2seq models produce high-dimensional vectors at each stage, e.g., encoder states, decoder states, or context states. It is difficult to interpret these vectors directly. However, we can estimate their meaning by looking at examples that produce similar vectors. To enable this comparison, we precompute the hidden states of a large set of example sentences (we use 50k sentences from the training set). For each state produced by the model on a given example, Seq2Seq-Vis searches for nearest neighbors from this large subset of precomputed states. These nearest neighbors are input to the *state trajectories* (Figure 6.7g) and to the *neighbor list* (Figure 6.7h).

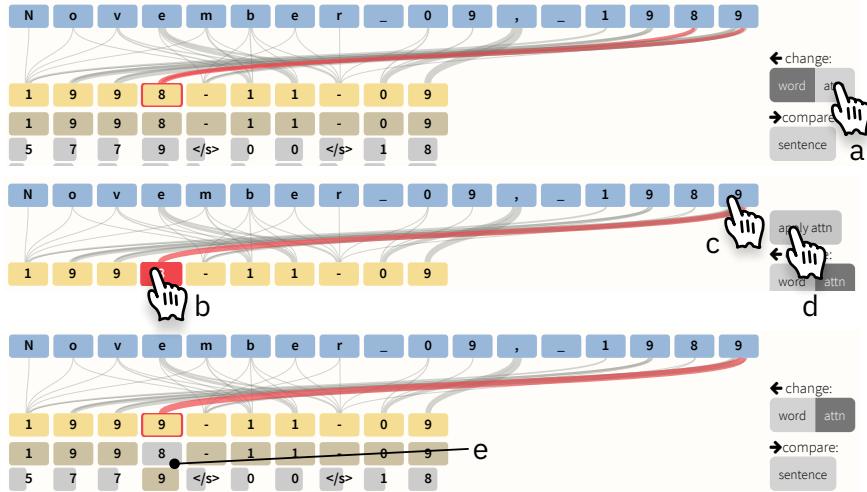
The state trajectories show the changing internal hidden state of the model with the goal of facilitating task  $T_2$ . This view encodes the dynamics of a model as a continuous trajectory. First, the set for

all states and their closest neighbors are projected using a non-linear algorithm, such as t-SNE (Maaten and Hinton, 2008), non-metric MDS (Kruskal, 1964), or a custom projection (see section 6.4). This gives a 2D positioning for each vector. We use these positions to represent each encoder/decoder sequence as a trace connecting its vectors. See Figure 6.7g for an example of a trace representing the encoder states for *wir wollen heute mal richtig spass haben*.

In the projection, the nearest neighbors to each vector are shown as nearby dots. When hovering over a vector from the input, the related nearest neighbor counterparts are highlighted and a temporary red line connects them. For vectors with many connections (high centrality), we reduce visual clutter by computing a concave hull for all related neighbors and highlight the related dots within the hull. Furthermore, we set the radius of each neighbor dot to be dependent on how many original states refer to it. For example, if three states from a decoder sequence have one common neighbor, the neighbor's radius is set to  $\sim 2.5$  (we use a  $r(x) = \sqrt{2x}$  mapping with  $x$  being number of common neighbors).

The state trajectories can be quite long. To ease understanding, we render a view showing states in their local neighborhood as a series of trajectory pictograms (Figure 6.7f). Each little window is a cut-out from the projection view, derived from applying a regular grid on top of the projection plane. Each pictogram shows only the cut-out region in which the respective vector can be found.

Clicking on any projected vector will show the neighbor list on the right side of the view. The neighbor list shows the actual sentences corresponding to the neighbor points. Specifically, the neighbor list shows all the nearest neighbors for the selected point with the original sequence pair. The source or target position in the sequence that matches is highlighted in red. The user can facet the list by filtering only to show source (blue) or target (yellow) sequences. She can also offset the text highlight by  $-1$  or  $+1$  to see alignment for preceding or succeeding word positions (see Figure 6.5).



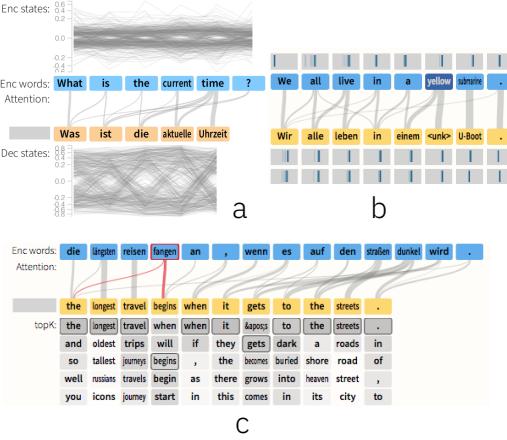
**Figure 6.8:** To re-direct attention in Seq2Seq-Vis, the user first observes a split of attention between the input 8 and 9 for converting the last digits of a year in a date conversion model. She can (a) select attention mode, (b) select the decoder word, (c) click on the preferred encoder word, (d) apply the attention change, and (e) see the models reaction.

### 6.3.1 INTERACTING WITH EXAMPLES

SEQ2SEQ-VIS allows multiple ways to interact with its different parts to facilitate analysis of the model. In order to gain an intuition into the model behavior and to test potential fixes, the interface needs to facilitate multiple different interactions.

One such interaction is to modify the model directly, for instance, by altering attention weights (\$3 in model). For this step, Beth can select a target word for which attention should be modified. By repeatedly clicking on encoder words, she gives more weights to these encoder words. Figure 6.8 shows how the attention can be modified for an example. After hitting *apply attn*, the attention is applied for this position, overwriting the original attention distribution.

Beth can further specify direct changes to either the source or the target. The user can trigger the changes by using the *manual compare* button and enter a new source or a new target sentence. When the source is changed, a new full translation is triggered. If the target is changed, a prefix decode is triggered. That means that the current prefix  $y_1, \dots, y_t$  is fixed during the beam search which then



**Figure 6.9:** Design variants for additional token information: (a) progression of hidden states, (b) density of neighborhoods, or (c) top  $K$  predictions as heatmap.

looks for the best sequence

$$\hat{\mathbf{y}} = \arg \max_{\mathbf{y} \in \mathcal{Y}} p(\mathbf{y} | \mathbf{x}, y_1, \dots, y_t).$$

Alternatively, as demonstrated in the use case, Beth can select the word from the top  $K$  predictions (Figure 6.7d) that seems to be the best next word. By clicking on one of these words, a prefix decode is triggered as described above.

### 6.3.2 DESIGN ITERATIONS

We considered several different variants for both main views of the system. For the translation view, we considered incorporating more state information directly into the encoding. Figure 6.9 shows iterations for adding per-word information around encoder and decoder. Similar to LSTMVis, the hidden state line charts show progression along encoder and decoder hidden states (Figure 6.9a). Domain scientists rejected this as too noisy for the given domain goals. In a later iteration the visualization experts proposed to indicate the closeness of the nearest neighbors with a simple histogram-like en-

coding (Figure 6.9b). This information did not help to formulate hypotheses and it failed to reveal a lot of variance. The next design focused on incorporating language features rather than latent vectors. It showed for each time step of the decoder the top  $K$  predicted words being produced as if there was only the top beam evaluated until then. In the final iteration, the only change was to change the indication of probability from a heatmap to a length-based encoding (Figure 6.7d).

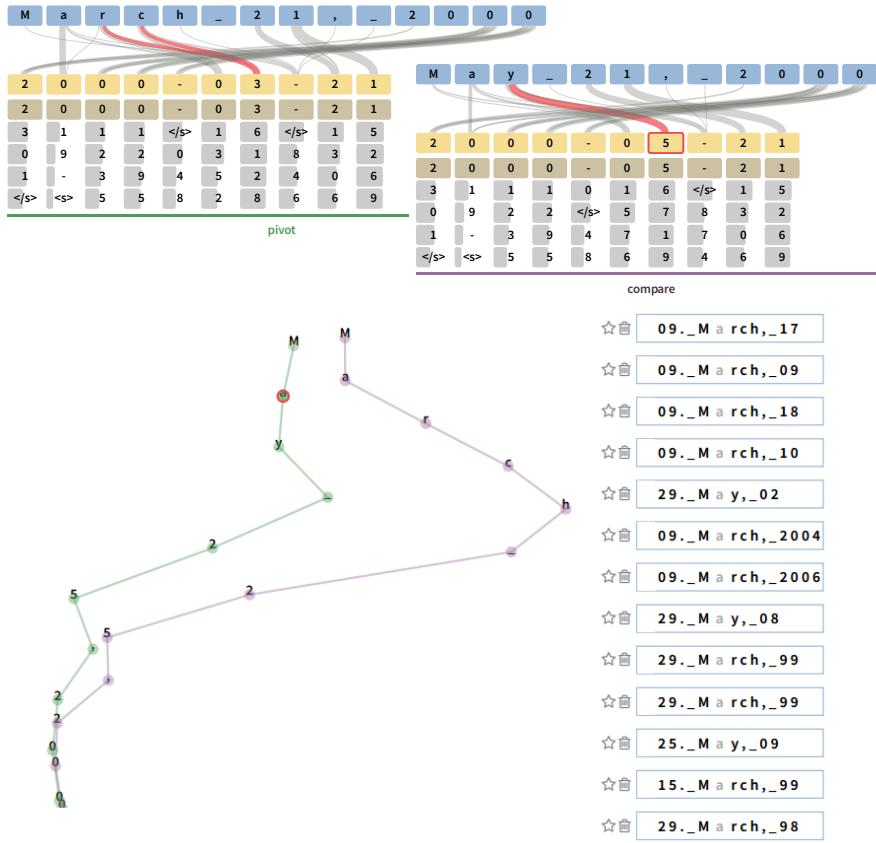
These iterations show that, while some visual encoding work for one use case, not all insights can be directly transferred. This further reinforces the need to develop the eventual users of a tool and to use their feedback to improve the system.

#### 6.4 USE CASES

We demonstrate the application of SEQ2SEQ-VIS and how it helps to generate insights using examples from a toy date conversion problem, abstractive summarization, and machine translation (section 6.1).

**Date Conversion.** Seq2seq models can be difficult to build and debug even for simple problems. A common test case used to check whether a model is implemented correctly is to learn a well-specified deterministic task. Here we consider the use case of converting various date formats to the unified format YEAR-MONTH-DAY. For example, the source *March 25, 2000* should be converted to the target *2000-03-25*. While this problem is much simpler than language translation, it tests the different components of the system. Specifically, the encoder (S1) must learn to identify different months, the attention (S3) must learn to reorder between the source and the target, and the decoder (S2) must express the source word in a numeric format.

Seq2Seq-Vis provides tools for examining these different stages of the model. Figure 6.10 shows an example, where the user, following Goal 3, employs a comparison between two different translations, one starting with *March* and the other with *May*. These two translations are nearly identical, except



**Figure 6.10:** Comparing translations for a date conversion model. The input sequences *March 21, 2000* and *May 21, 2000* are only different by some letters. The attention (top) for predicting the correct months 3 and 5 is focused on this difference (*y* vs. *rc*). The trajectory view (bottom left) shows this difference along the progression of encoder states. The neighborhood list (bottom right) indicates that after input of *M a* the model is still undecided.

one yields the month 3 and the other 5. Following Goal 1, the user might want to examine the models decisions. The upper translation view provides a way to compare between the attention on the two inputs. The red highlighted connections indicate that the first sentence attention focuses on *rc* whereas the second focuses on *y*. These characters are used by the model to distinguish the two months since it cannot use *M a*. The user can also observe how the encoder learns to use these letters. The trajectory view compares the encoder states of sentence 1 and sentence 2. Here we use a custom projection, where the y-axis is the relative position of a word in a sentence and the x-axis is a 1-d projection of the

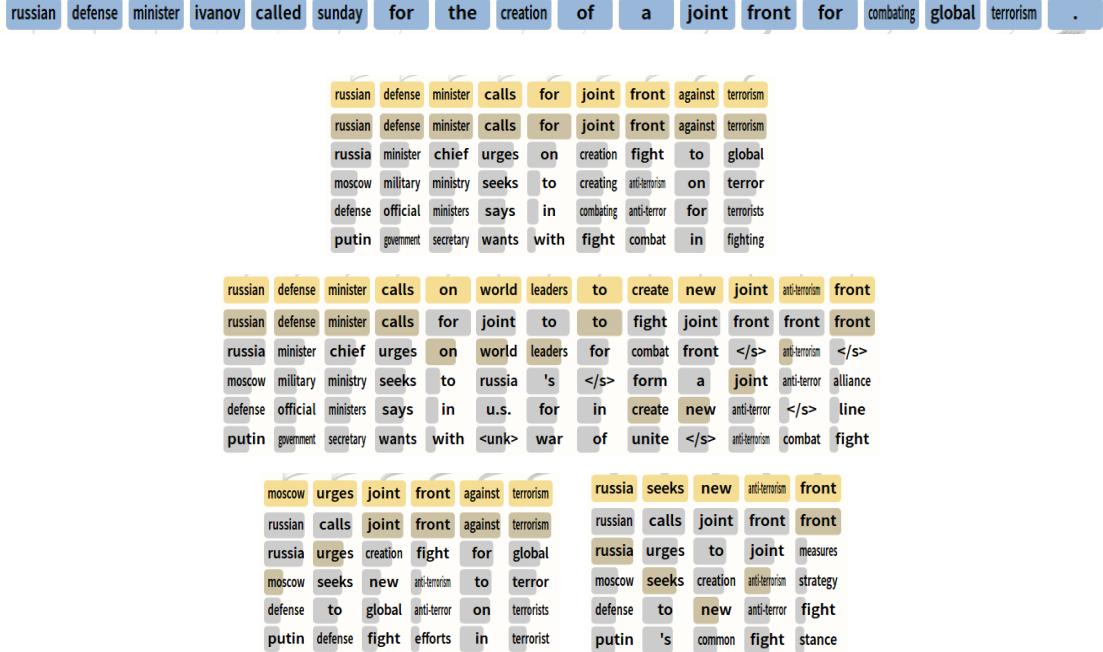
vector. This reveals that the two trajectories are similar before and after these characters, but diverge significantly around  $r$  and  $c$ . Finally, following Goal 2, the user can connect these decisions back to the training data. On the right, she can see the nearest neighbors around the letter  $a$  in  $M a y$  (highlighted in red). Interestingly, the set of nearest neighbors is almost equally split between examples of  $M a y$  and  $M a r c h$ , indicating that at this stage of decoding the model is preserving its uncertainty between the two months.

**Abstractive Summarization.** For our second use case we apply the tool to a summarization problem. Recently, researchers have developed methods for *abstractive* text summarization that learn how to produce a shorter summarized version of a text passage. Seq2seq models are commonly used in this framework (Rush et al., 2015, Nallapati et al., 2016b, Paulus et al., 2018, See et al., 2017). In abstractive summarization, the target passage may not contain the same phrasing as the original. Instead, the model learns to paraphrase and alter the wording in the process of summarization.

Studying how paraphrasing happens in seq2seq systems is a core research question in this area. Rush et al. (2015) describe a system using the Gigaword data set (3.8M sentences). They study the example source sentence *russian defense minister ivanov called sunday for the creation of a joint front for combating global terrorism* to produce a summary *russia calls for joint front against terrorism*. Here *russia* compresses the phrase *russian defense minister ivanov* and *against* paraphrases *for combating*.

To replicate this use case we consider a user Bob analyzing this sentence. In particular, he is interested in understanding how the model selects the length and the level of abstraction. Bob can analyze this in the context of Goal 3, testing alternatives predictions of the model, in particular targeting Stage 4. As discussed in Sect 6.3, Seq2Seq-Vis shows the top  $K$  predictions at each time step. When the user clicks on a prediction, the system will produce a sentence that incorporates this prediction. Each choice is “locked” so that further alterations can be made.

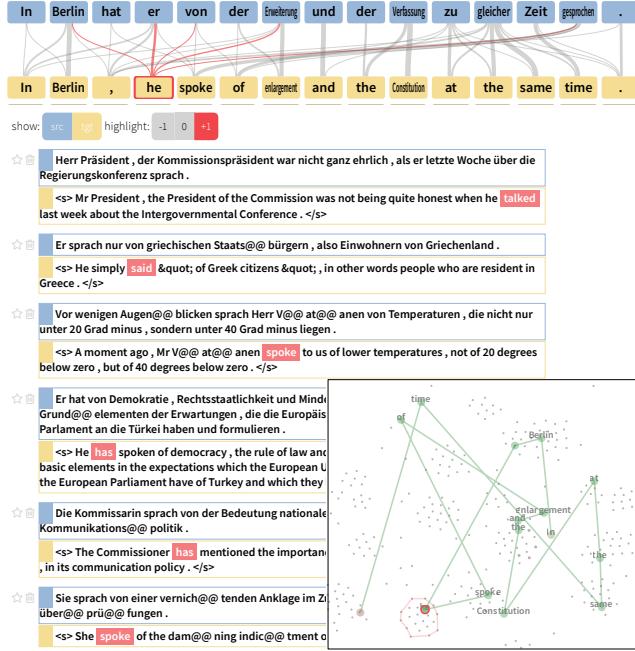
Figure 6.11 shows the source input to this model. We can see four different summaries that the model produces based on different word choices. Interestingly, specific local choices do have a sig-



**Figure 6.11:** Use case of abstractive summarization. The input sentence *russian defense minister ivanov called sunday for the creation of a joint front for combating global terrorism* can be summarized in different ways. The yellow boxes indicate alternative translations for different prefix decode settings. Top: the unconstrained abstraction; middle: changing prediction from *for* to *on* leads to automatic insertion of *on world leaders* to stay grammatically correct; bottom left: changing the first word from *russian* to *moscow* or *russia* compresses the sentence even more while retaining its meaning.

nificant impact on length, ranging from five to thirteen words. By switching from *for* to *on*, Bob can lead the decoder to insert an additional phrase *on world leaders* to maintain grammaticality. While the model outputs the top choice, all other choices have relatively high probabilities. This observation has motivated research into adding constraints to the prediction at each time step. Consequently, we have added methods for constraining length and prediction into the underlying seq2seq system to produce different outputs.

**Machine Translation.** Finally, we consider a more in-depth use case of a real-world machine translation system using a large model trained on WMT ’14 (3.96M examples) to translate from German to English. This use case considers a holistic view of how an expert might go about understanding the



**Figure 6.12:** Use case language translation using WMT'14 data. The attention graph (top) shows how attention for the target word *he* is not only focused on the decoder counterpart *er* but also on the following words, even to the far away verb *gesprochen* (spoke). The state trajectory for the decoder states reveals how close *he* and *spoke* are. The neighborhood list indicates that the model sets the stage for predicting *spoke* as next word.

decisions of the system.

Figure 6.12 shows an example source input and its translation. Here the user has input a source sentence, translated it, and activated the neighbor view to consider the decoder states. She is interested in better understanding each stage of the model at this point. This sentence is interesting as there is significant reordering that must occur to translate from the original German to English. For instance, the subject *he* is at the beginning of the clause, but must interact with the verb *gesprochen* at the end of the German sentence.

We consider Goals 1 and 2 applied to this example, with the intent of analyzing the encoder, decoder, attention, and prediction (S1-S4). First we look at the attention. Normally, this stage focuses on the word it is translating (*er*), but researchers have noted that neural models often look ahead to



**Figure 6.13:** An under-trained English-German model. Repetition repetition is a commonly observed phenomenon in under-trained or under-parametrized models. Here the trajectory pictograms show that for the repetition *in Stuttgart in Stuttgart* the decoder states alternate in the same region before being able to break apart.

the next word in this process (Koehn and Knowles, 2017). We can see branches going from *he* to potential next steps (e.g., *von* or *gesprochen*). We can further view this process in the decoder trajectory shown below, where *he* and *speak* are placed near each other in the path. Hovering over the vector *he* highlights it globally in the tool. Furthermore, if we click on *he*, we can link this state to other examples in our data (Goal 2). On the right we can see these related examples, with the next word (+1) highlighted. We find that the decoder is representing not just the information for the current word, but also anticipating the translation of the verb *sprechen* in various forms.

In this case we are seeing the model behaving correctly to produce a good translation. However, the tool can also be useful when there are issues with the system. One common issue in under-trained or under-parameterized seq2seq models is to repeatedly generate the same phrase. Figure 6.13 shows an example of this happening. The model repeats the phrase *in Stuttgart in Stuttgart*. We can observe in the pictogram view that the decoder model has produced a loop, ending up in nearly the same position even after seeing the next word. As a short-term fix, the tool’s prefix decoding can avoid this issue. It remains an interesting research question to prevent this type of cycle from occurring.

## 6.5 CONCLUSIONS

We have presented SEQ2SEQ-VIS, a tool to facilitate the exploration of all stages of a seq2seq model. We apply our set of goals to deep learning models that are traditionally difficult to interpret. This tool is the first of its kind to combine insights about model mechanics (translation view) with in-

sights about model semantics (neighborhood view), while allowing for “what if”-style counterfactual changes of the model’s internals. The interaction with the model internals, in addition to constraining the predictions and changing the inputs, leads to a better understanding of how the model performs. Moreover, it allows a user to develop a mental model of the model behavior so that they can learn to recognize the limitations of a model.

However, this style of output-driven exploration is limited by the number of controllable parts of a model. In the case of `SEQ2SEQ-VIS`, the attention is the only model-internal that is easy to understand and manipulate. We thus pose that model architectures need to be extended with further understandable components to achieve a higher degree of coupling between interface and model.

*The theory of probabilities is at bottom nothing  
but common sense reduced to calculus.*

Pierre-Simon Laplace

# 7

## Bottom-Up Summarization: Extending Models with Controllable Variables

The previous case studies demonstrated the need for interactive systems, but were also limited by the number of possible interactions with a model. This became especially clear in SEQ2SEQ-V1s where all the interactions are limited to those that are (1) inherently understandable by the user of the tool, and (2) available within the model. To extend the possible design space for interactions, we thus need to

extend models with facets that are both understandable and can be used to control the model reasoning process. One approach to this problem, which we demonstrate in this chapter, is to consider how humans address an NLG tasks, in this case abstractive summarization. Specifically, we show how to modify the model to follow the human-like domain-specific reasoning process.

As described in Chapter 2.2, text summarization systems aim to generate natural language summaries that compress the information in a longer text. Approaches using neural networks have shown promising results on this task with end-to-end models that encode a source document and then decode it into an abstractive summary. The most commonly applied neural abstractive summarization models combine extractive and abstractive techniques by using pointer-generator style models which can copy words from the source document (Gu et al., 2016, See et al., 2017). These end-to-end models produce fluent abstractive summaries but have had mixed success in content selection, i.e. deciding what to summarize, compared to fully extractive models.

There is an appeal to end-to-end models from a modeling perspective; however, there is evidence that when summarizing people follow a two-step approach of first selecting important phrases and then paraphrasing them (Anderson and Hidi, 1988, Jing and McKeown, 1999). A similar argument has been made for image captioning. Anderson et al. (2018) develop a state-of-the-art model with a two-step approach that first pre-computes bounding boxes of segmented objects and then applies attention to these regions. This so-called *bottom-up* attention is inspired by neuroscience research describing attention based on properties inherent to a stimulus (Buschman and Miller, 2007).

Motivated by this approach, we consider *bottom-up* attention for neural abstractive summarization. Our approach first selects a selection mask for the source document and then constrains a standard neural model by this mask. This approach can better decide which phrases a model should include in a summary, without sacrificing the fluency advantages of neural abstractive summarizers. Furthermore, it requires much fewer data to train, which makes it more adaptable to new domains.

Our full model incorporates a separate content selection system to decide on relevant aspects of the

### Source Document

german chancellor angela merkel [did] not [look] too pleased about the weather during her [annual] easter holiday [in italy.] as britain [basks] in [sunshine] and temperatures of up to 21c, mrs merkel and her husband[, chemistry professor joachim sauer,] had to settle for a measly 12 degrees. the chancellor and her [spouse] have been spending easter on the small island of ischia, near naples in the mediterranean for over a [decade.]

[not so sunny:] angela merkel [and] her husband[, chemistry professor joachim sauer,] are spotted on their [annual] easter trip to the island of ischia[,] near naples[. the] couple [traditionally] spend their holiday at the five-star miramare spa hotel on the south of the island [, which comes] with its own private beach [, and balconies overlooking the] ocean [.]...

### Reference

- angela merkel and husband spotted while on italian island holiday.

...

### Baseline Approach

- angela merkel and her husband, chemistry professor joachim sauer, are spotted on their annual easter trip to the island of ischia, near naples.

...

### Bottom-Up Summarization

- angela merkel and her husband are spotted on their easter trip to the island of ischia, near naples.

...

**Figure 7.1:** Example of two sentence summaries with and without bottom-up attention. The model does not allow copying of words in [gray], although it can generate words. With bottom-up attention, we see more explicit sentence compression, while without it whole sentences are copied verbatim.

source document. We frame this selection task as a sequence-tagging problem, with the objective of identifying tokens from a document that are part of its summary. We show that a content selection model that builds on contextual word embeddings (Peters et al., 2018) can identify correct tokens with a recall of over 60%, and a precision of over 50%. To incorporate bottom-up attention into abstractive summarization models, we employ masking to constrain copying words to the selected parts of the text, which produces grammatical outputs. We additionally experiment with multiple methods to incorporate similar constraints into the training process of more complex end-to-end abstractive summarization models, either through multi-task learning or through directly incorporating a fully differentiable mask.

Our experiments compare bottom-up attention with several other state-of-the-art abstractive systems. Compared to our baseline models of See et al. (2017) bottom-up attention leads to an improvement in ROUGE-L score on the CNN-DM corpus from 36.4 to 38.3 while being simpler to train. We also see comparable or better results than recent reinforcement-learning based methods with our MLE trained system. Furthermore, we find that the content selection model is very data-efficient and can be trained with less than 1% of the original training data. This provides opportunities for domain-transfer and low-resource summarization. We show that a summarization model trained on CNN-DM and evaluated on the NYT corpus can be improved by over 5 points in ROUGE-L with a content selector trained on only 1,000 in-domain sentences.

## 7.1 RELATED WORK

There is a tension in document summarization between staying close to the source document and allowing compressive or abstractive modification. Many non-neural systems take a select and compress approach. For example, Dorr et al. (2003) introduced a system that first extracts noun and verb phrases from the first sentence of a news article and uses an iterative shortening algorithm to compress it. Recent systems such as Durrett et al. (2016) also learn a model to select sentences and then compress them.

In contrast, recent work in neural network based data-driven extractive summarization has focused on extracting and ordering full sentences (Cheng and Lapata, 2016, Dlikman and Last, 2016). Nallapati et al. (2016a) use a classifier to determine whether to include a sentence and a selector that ranks the positively classified ones. These methods often over-extract, but extraction at a word level requires maintaining grammatically correct output (Cheng and Lapata, 2016), which is difficult. Interestingly, key phrase extraction while ungrammatical often matches closely in content with human-generated summaries (Bui et al., 2016).

A third approach is neural abstractive summarization with sequence-to-sequence models (Sutskever et al., 2014, Bahdanau et al., 2015). These methods have been applied to tasks such as headline generation (Rush et al., 2015) and article summarization (Nallapati et al., 2016b). Chopra et al. (2016) show that attention approaches that are more specific to summarization can further improve the performance of models. Gu et al. (2016) were the first to show that a copy mechanism, introduced by Vinyals et al. (2015), can combine the advantages of both extractive and abstractive summarization by copying words from the source. See et al. (2017) refine this pointer-generator approach and use an additional coverage mechanism (Tu et al., 2016) that makes a model aware of its attention history to prevent repeated attention.

Most recently, reinforcement learning (RL) approaches that optimize objectives for summarization other than maximum likelihood have been shown to further improve performance on these tasks (Paulus et al., 2018, Li et al., 2018b, Celikyilmaz et al., 2018). Paulus et al. (2018) approach the coverage problem with an intra-attention in which a decoder has an attention over previously generated words. However RL-based training can be difficult to tune and slow to train. Our method does not utilize RL training, although in theory this approach can be adapted to RL methods.

Several papers also explore multi-pass extractive-abstractive summarization. Nallapati et al. (2017) create a new source document comprised of the important sentences from the source and then train an abstractive system. Liu\* et al. (2018) describe an extractive phase that extracts full paragraphs and an abstractive one that determines their order. Finally Zeng et al. (2016) introduce a mechanism that reads a source document in two passes and uses the information from the first pass to bias the second. Our method differs in that we utilize a completely abstractive model, biased with a powerful content selector.

Other recent work explores alternative approaches to content selection. For example, Cohan et al. (2018) use a hierarchical attention to detect relevant sections in a document, Li et al. (2018a) generate a set of keywords that is used to guide the summarization process, and Pasunuru and Bansal

(2018) develop a loss-function based on whether salient keywords are included in a summary. Other approaches investigate the content-selection at the sentence-level. Tan et al. (2017) describe a graph-based attention to attend to one sentence at a time, Chen and Bansal (2018) first extract full sentences from a document and then compress them, and Hsu et al. (2018) modulate the attention based on how likely a sentence is included in a summary.

## 7.2 BACKGROUND: NEURAL SUMMARIZATION

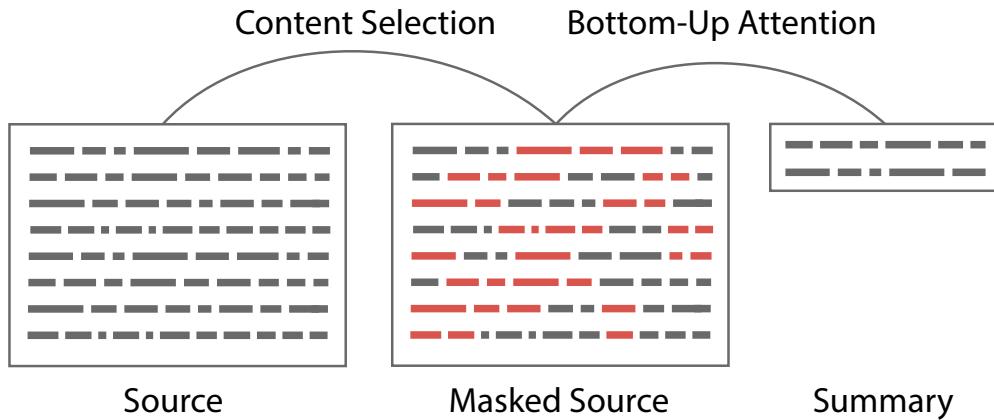
Throughout this paper, we consider a set of pairs of texts  $(\mathcal{X}, \mathcal{Y})$  where  $\mathbf{x} \in \mathcal{X}$  corresponds to source tokens  $x_1, \dots, x_S$  and  $\mathbf{y} \in \mathcal{Y}$  to a summary  $y_1, \dots, y_T$  with  $T \ll S$ .

Abstractive summaries are generated one word at a time. At every time-step, a model is aware of the previously generated words. The problem is to learn a function  $f(\mathbf{x})$  parametrized by  $\theta$  that maximizes the probability of generating the correct sequences. Following previous work, we model the abstractive summarization with an attentional sequence-to-sequence model. The attention distribution  $p(\mathbf{a}_t | \mathbf{x}, \mathbf{y}_{1:t})$  for a decoding step  $t$ , calculated within the neural network, represents an embedded soft distribution over all of the source tokens and can be interpreted as the current focus of the model.

The model additionally has a copy mechanism (Vinyals et al., 2015) to copy words from the source. Copy models extend the decoder by predicting a binary soft switch  $z_t$  that determines whether the model copies or generates. The copy distribution is a probability distribution over the source text, and the joint distribution is computed as a convex combination of the two parts of the model,

$$\begin{aligned} p(y_{t+1} | \mathbf{y}_{1:t}, \mathbf{x}) = \\ p(z_t = 1 | \mathbf{y}_{1:t}, \mathbf{x}) \times p(y_{t+1} | z_t = 1, \mathbf{y}_{1:t}, \mathbf{x}) + \\ p(z_t = 0 | \mathbf{y}_{1:t}, \mathbf{x}) \times p(y_{t+1} | z_t = 0, \mathbf{y}_{1:t}, \mathbf{x}) \end{aligned} \tag{7.1}$$

where the two parts represent copy and generation distribution respectively. Following the *pointer-*



**Figure 7.2:** Overview of the selection and generation processes described throughout Section 7.3.

*generator* model of See et al. (2017), we reuse the attention  $p(\mathbf{a}_t | \mathbf{x}, \mathbf{y}_{1:t})$  distribution as copy distribution, i.e. the copy probability of a token in the source  $w$  through the copy attention is computed as the sum of attention towards all occurrences of  $w$ . During training, we maximize marginal likelihood with the latent switch variable, such that

$$p(\mathbf{y}_{t+1} = w | z_j = 1, \mathbf{y}_{1:t}, \mathbf{x}) = \sum_{s: w_s = w} a_t^s.$$

### 7.3 BOTTOM-UP ATTENTION

We next consider techniques for incorporating a content selection into abstractive summarization, illustrated in Figure 7.2.

### 7.3.1 CONTENT SELECTION

We define the content selection problem as a word-level extractive summarization task. While there has been significant work on custom extractive summarization, we make a simplifying assumption and treat it as a sequence tagging problem. Let  $t_1, \dots, t_S$  denote binary tags for each of the source tokens, i.e. 1 if a word is copied in the target sequence and 0 otherwise.

While there is no supervised data for this task, we can generate training data by aligning the summaries to the document. We define a word  $x_i$  as copied if (1) it is part of the longest possible subsequence of tokens  $s = x_{i-j:i:i+k}$ , for integers  $j \leq i; k \leq (n - i)$ , if  $s \in x$  and  $s \in y$ , and (2) there exists no earlier sequence  $u$  with  $s = u$ .

We use a standard bidirectional LSTM model trained with maximum likelihood for the sequence labeling problem. Recent results have shown that better word representations can lead to significantly improved performance in sequence tagging tasks (Peters et al., 2017). Therefore, we first map each token  $w_i$  into two embedding channels  $\mathbf{e}_i^{(w)}$  and  $\mathbf{e}_i^{(c)}$ . The  $\mathbf{e}^{(w)}$  embedding represents a static channel of pre-trained word embeddings, e.g. GLoVe (Pennington et al., 2014). The  $\mathbf{e}^{(c)}$  are contextual embeddings from a pretrained language model, e.g. ELMo (Peters et al., 2018) which uses a character-aware token embedding (Kim et al., 2016b) followed by two bidirectional LSTM layers  $\mathbf{h}_i^{(1)}$  and  $\mathbf{h}_i^{(2)}$ . The contextual embeddings are fine-tuned to learn a task-specific embedding  $\mathbf{e}_i^{(c)}$  as a linear combination of the states of each LSTM layer and the token embedding,

$$\mathbf{e}_i^{(c)} = \gamma \times \sum_{\ell=0}^2 s_j \times \mathbf{h}_i^{(\ell)},$$

with  $\gamma$  and  $s_{0,1,2}$  as trainable parameters. Since these embeddings only add four additional parameters to the tagger, it remains very data-efficient despite the high-dimensional embedding space.

Both embeddings are concatenated into a single vector that is used as input to a bidirectional LSTM, which computes a representation  $\mathbf{h}_s$  for a word  $x_s$ . We can then calculate the probability  $q_s$  that the

word is selected as  $\sigma(\mathbf{W}_c h_s + \mathbf{b}_c)$  with trainable parameters  $\mathbf{W}_c$  and  $\mathbf{b}_c$ .

### 7.3.2 BOTTOM-UP COPY ATTENTION

Inspired by work in bottom-up attention for images (Anderson et al., 2018) which restricts attention to predetermined bounding boxes within an image, we use these attention masks to limit the available selection of the pointer-generator model.

As shown in Figure 7.1, a common mistake made by neural copy models is copying very long sequences or even whole sentences. In the baseline model, over 50% of copied tokens are part of copy sequences that are longer than 10 tokens, whereas this number is only 10% for reference summaries. While bottom-up attention could also be used to modify the source encoder representations, we found that a standard encoder over the full text was effective at aggregation and therefore limit the bottom-up step to attention masking.

Concretely, we first train a pointer-generator model on the full dataset as well as the content selector defined above. At inference time, to generate the mask, the content selector computes selection probabilities  $q_{1:S}$  for each token in a source document. The selection probabilities are used to modify the copy attention distribution to only include tokens identified by the selector. Let  $a_t^s$  denote the attention at decoding step  $t$  to the encoder word at time step  $s$ . Given a threshold  $\epsilon$ , the selection is applied as a hard mask, such that

$$p(\tilde{a}_t^s | \mathbf{x}, \mathbf{y}_{1:t}) = \begin{cases} p(a_t^s | \mathbf{x}, \mathbf{y}_{1:t}) & q_s > \epsilon \\ 0 & \text{ow.} \end{cases}$$

To ensure that Eq. 7.1 still yields a correct probability distribution, we first multiply  $p(\tilde{a}_t^s | \mathbf{x}, \mathbf{y}_{1:t})$  by a normalization parameter  $\lambda^1$  and then renormalize the distribution. The resulting normalized distri-

---

<sup>1</sup>Empirically, we found that values around  $\lambda = 2$  yield the best results.

bution can be used to directly replace  $a$  as the new copy probabilities.

### 7.3.3 END-TO-END ALTERNATIVES

Two-step BOTTOM-UP attention has the advantage of training simplicity. In theory, though, standard copy attention should be able to learn how to perform content selection as part of the end-to-end training. We consider several other end-to-end approaches for incorporating content selection into neural training.

Method 1: (MASK ONLY): We first consider whether the alignment used in the bottom-up approach could help a standard summarization system. Inspired by Nallapati et al. (2017), we investigate whether aligning the summary and the source during training and fixing the gold copy attention to pick the "correct" source word is beneficial. We can think of this approach as limiting the set of possible copies to a fixed source word. Here the training is changed, but no mask is used at test time.

Method 2 (MULTI-TASK): Next, we investigate whether the content selector can be trained alongside the abstractive system. We first test this hypothesis by posing summarization as a multi-task problem and training the tagger and summarization model with the same features. For this setup, we use a shared encoder for both abstractive summarization and content selection. At test time, we apply the same masking method as bottom-up attention.

Method 3 (DIFFMASK): Finally we consider training the full system end-to-end with the mask during training. Here we jointly optimize both objectives, but use predicted selection probabilities to softly mask the copy attention  $p(\tilde{\mathbf{a}}_t^s | \mathbf{x}, \mathbf{y}_{1:t}) = p(a_t^s | \mathbf{x}, \mathbf{y}_{1:t}) \times q_s$ , which leads to a fully differentiable model. This model is used with the same soft mask at test time.

## 7.4 INFERENCE

Several authors have noted that longer-form neural generation still has significant issues with incorrect length and repeated words than in short-form problems like translation. Proposed solutions include modifying models with extensions such as a coverage mechanism (Tu et al., 2016, See et al., 2017) or intra-sentence attention (Cheng et al., 2016, Paulus et al., 2018). We instead stick to the theme of modifying inference, and modify the scoring function to include a length penalty  $lp$  and a coverage penalty  $cp$ , and is defined as  $s(\mathbf{x}, \mathbf{y}) = \log p(\mathbf{y}|\mathbf{x})/lp(\mathbf{x}) - cp(\mathbf{x}; \mathbf{y})$ .

*Length:* To encourage the generation of longer sequences, we apply length normalizations during beam search. We use the length penalty by Wu et al. (2016), which is formulated as

$$lp(\mathbf{y}) = \frac{(5 + |\mathbf{y}|)^\alpha}{(5 + 1)^\alpha},$$

with a tunable parameter  $\alpha$ , where increasing  $\alpha$  leads to longer summaries. We additionally set a minimum length based on the training data.

*Repeats:* Copy models often repeatedly attend to the same source tokens, generating the same phrase multiple times. We introduce a new summary specific coverage penalty,

$$cp(\mathbf{x}; \mathbf{y}) = \beta \left( -T + \sum_{t=1}^T \max \left( 1.0, \sum_{s=1}^S a_t^s \right) \right).$$

Intuitively, this penalty increases whenever the decoder directs more than 1.0 of total attention within a sequence towards a single encoded token. By selecting a sufficiently high  $\beta$ , this penalty blocks summaries whenever they would lead to repetitions. Additionally, we follow (Paulus et al., 2018) and restrict the beam search to never repeat trigrams.

## 7.5 DATA AND EXPERIMENTS

We evaluate our approach on the CNN-DM corpus (Hermann et al., 2015, Nallapati et al., 2016b), and the NYT corpus (Sandhaus, 2008), which are both standard corpora for news summarization. The summaries for the CNN-DM corpus are bullet points for the articles shown on their respective websites, whereas the NYT corpus contains summaries written by library scientists. CNN-DM summaries are full sentences, with on average 66 tokens ( $\sigma = 26$ ) and 4.9 bullet points. NYT summaries are not always complete sentences and are shorter, with on average 40 tokens ( $\sigma = 27$ ) and 1.9 bullet points. Following See et al. (2017), we use the non-anonymized version of the CNN-DM corpus and truncate source documents to 400 tokens and the target summaries to 100 tokens in training and validation sets. For experiments with the NYT corpus, we use the preprocessing described by Paulus et al. (2018), and additionally remove author information and truncate source documents to 400 tokens instead of 800. These changes lead to an average of 326 tokens per article, a decrease from the 549 tokens with 800 token truncated articles. The target (non-copy) vocabulary is limited to 50,000 tokens for all models.

The content selection model uses pre-trained GloVe embeddings of size 100, and ELMo with size 1024. The biLSTM has two layers and a hidden size of 256. Dropout is set to 0.5, and the model is trained with Adagrad, an initial learning rate of 0.15, and an initial accumulator value of 0.1. We limit the number of training examples to 100,000 on either corpus, which only has a small impact on performance. For the jointly trained content selection models, we use the same configuration as the abstractive model.

For the base model, we re-implemented the Pointer-Generator model as described by See et al. (2017). To have a comparable number of parameters to previous work, we use an encoder with 256 hidden states for both directions in the one-layer LSTM, and 512 for the one-layer decoder. The embedding size is set to 128. The model is trained with the same Adagrad configuration as the content

selector. Additionally, the learning rate halves after each epoch once the validation perplexity does not decrease after an epoch. We do not use dropout and use gradient-clipping with a maximum norm of 2. We found that increasing model size or using the Transformer (Vaswani et al., 2017) can lead to slightly improved performance, but at the cost of increased training time and parameters. We report numbers of a Transformer with copy-attention, which we denote CopyTransformer. In this model, we randomly choose one of the attention-heads as the copy-distribution, and otherwise follow the parameters of the big Transformer by Vaswani et al. (2017).

All inference parameters are tuned on a 200 example subset of the validation set. Length penalty parameter  $\alpha$  and copy mask  $\epsilon$  differ across models, with  $\alpha$  ranging from 0.6 to 1.4, and  $\epsilon$  ranging from 0.1 to 0.2. The minimum length of the generated summary is set to 35 for CNN-DM and 6 for NYT. While the Pointer-Generator uses a beam size of 5 and does not improve with a larger beam, we found that bottom-up attention requires a larger beam size of 10. The coverage penalty parameter  $\beta$  is set to 10, and the copy attention normalization parameter  $\lambda$  to 2 for both approaches. We use AllenNLP (Gardner et al., 2018) for the content selector, and OpenNMT-py for the abstractive models (Klein et al., 2017).<sup>2</sup>.

## 7.6 RESULTS

Table 7.1 shows our main results on the CNN-DM corpus, with abstractive models shown in the top, and bottom-up attention methods at the bottom. We first observe that using a coverage inference penalty scores the same as a full coverage mechanism, without requiring any additional model parameters or model fine-tuning. The results with the CopyTransformer and coverage penalty in-

---

<sup>2</sup>Code and reproduction instructions can be found at <https://github.com/sebastianGehrmann/bottom-up-summary>

<sup>2</sup>These results compare on the non-anonymized version of this corpus used by (See et al., 2017). The best results on the anonymized version are R1:41.69 R2:19.47 RL:37.92 from (Celikyilmaz et al., 2018). We compare to their DCA model on the NYT corpus.

Method	R-1	R-2	R-L
Pointer-Generator ( <a href="#">See et al., 2017</a> )	36.44	15.66	33.42
Pointer-Generator + Coverage ( <a href="#">See et al., 2017</a> )	39.53	17.28	36.38
ML + Intra-Attention ( <a href="#">Paulus et al., 2018</a> )	38.30	14.81	35.49
ML + RL ( <a href="#">Paulus et al., 2018</a> )	39.87	15.82	36.90
Saliency + Entailment reward ( <a href="#">Pasunuru and Bansal, 2018</a> )	40.43	18.00	37.10
Key information guide network ( <a href="#">Li et al., 2018a</a> )	38.95	17.12	35.68
Inconsistency loss ( <a href="#">Hsu et al., 2018</a> )	40.68	17.97	37.13
Sentence Rewriting ( <a href="#">Chen and Bansal, 2018</a> )	40.88	17.80	<b>38.54</b>
Pointer-Generator (our implementation)	36.25	16.17	33.41
Pointer-Generator + Coverage Penalty	39.12	17.35	36.12
CopyTransformer + Coverage Penalty	39.25	17.54	36.45
Pointer-Generator + Mask Only	37.70	15.63	35.49
Pointer-Generator + Multi-Task	37.67	15.59	35.47
Pointer-Generator + DiffMask	38.45	16.88	35.81
Bottom-Up Summarization	<b>41.22</b>	<b>18.68</b>	38.34
Bottom-Up Summarization (CopyTransformer)	40.96	18.38	38.16

**Table 7.1:** Results of abstractive summarizers on the CNN-DM dataset.<sup>3</sup> The first section shows encoder-decoder abstractive baselines trained with cross-entropy. The second section describes reinforcement-learning based approaches. The third section presents our baselines and the attention masking methods described in this work.

dicate a slight improvement across all three scores, but we observe no significant difference between Pointer-Generator and CopyTransformer with bottom-up attention.

We found that none of our end-to-end models lead to improvements, indicating that it is difficult to apply the masking during training without hurting the training process. The *Mask Only* model with increased supervision on the copy mechanism performs very similar to the *Multi-Task* model. On the other hand, bottom-up attention leads to a major improvement across all three scores. While we would expect better content selection to primarily improve ROUGE-1, the fact all three increase hints that the fluency is not being hurt specifically. Our cross-entropy trained approach even outperforms all of the reinforcement-learning based approaches in ROUGE-1 and 2, while the highest reported ROUGE-L score by [Chen and Bansal \(2018\)](#) falls within the 95% confidence interval of our results.

Table 7.2 shows experiments with the same systems on the NYT corpus. We see that the 2 point improvement compared to the baseline Pointer-Generator maximum-likelihood approach carries over

Method	R-1	R-2	R-L
ML*	44.26	27.43	40.41
ML+RL*	47.03	30.72	<b>43.10</b>
DCA†	<b>48.08</b>	31.19	42.33
Point.Gen. + Coverage Pen.	45.13	30.13	39.67
Bottom-Up Summarization	47.38	<b>31.23</b>	41.81

**Table 7.2:** Results on the NYT corpus, where we compare to RL trained models. \* marks models and results by [Paulus et al. \(2018\)](#), and † results by [Celikyilmaz et al. \(2018\)](#).

to this dataset. Here, the model outperforms the RL based model by [Paulus et al. \(2018\)](#) in ROUGE-1 and 2, but not L, and is comparable to the results of ([Celikyilmaz et al., 2018](#)) except for ROUGE-L. The same can be observed when comparing ML and our Pointer-Generator. We suspect that a difference in summary lengths due to our inference parameter choices leads to this difference, but did not have access to their models or summaries to investigate this claim. This shows that a bottom-up approach achieves competitive results even to models that are trained on summary-specific objectives.

The main benefit of bottom-up summarization seems to be from the reduction of mistakenly copied words. With the best Pointer-Generator models, the precision of copied words is 50.0% compared to the reference. This precision increases to 52.8%, which mostly drives the increase in R1. An independent-samples t-test shows that this improvement is statistically significant with  $t=14.7$  ( $p < 10^{-5}$ ). We also observe a decrease in average sentence length of summaries from 13 to 12 words when adding content selection compared to the Pointer-Generator while holding all other inference parameters constant.

**DOMAIN TRANSFER** While end-to-end training has become common, there are benefits to a two-step method. Since the content selector only needs to solve a binary tagging problem with pretrained vectors, it performs well even with very limited training data. As shown in Figure 7.3, with only 1,000 sentences, the model achieves an AUC of over 74. Beyond that size, the AUC of the model increases only slightly with increasing training data.

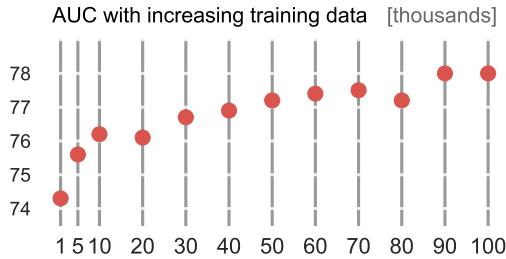


Figure 7.3: The AUC of the content selector trained on CNN-DM with different training set sizes ranging from 1,000 to 100,000 data points.

	AUC	R-1	R-2	R-L
CNNMDM	25.63	11.40	20.55	
+1k	80.7	30.62	16.10	25.32
+10k	83.6	32.07	17.60	26.75
+100k	86.6	33.11	18.57	27.69

Table 7.3: Results of the domain transfer experiment. AUC numbers are shown for content selectors. ROUGE scores represent an abstractive model trained on CNN-DM and evaluated on NYT, with additional copy constraints trained on 1/10/100k training examples of the NYT corpus.

To further evaluate the content selection, we consider an application to domain transfer. In this experiment, we apply the Pointer-Generator trained on CNN-DM to the NYT corpus. In addition, we train three content selectors on 1, 10, and 100 thousand sentences of the NYT set, and use these in the bottom-up summarization. The results, shown in Table 7.3, demonstrates that even a model trained on the smallest subset leads to an improvement of almost 5 points over the model without bottom-up attention. This improvement increases with the larger subsets to up to 7 points. While this approach does not reach a comparable performance to models trained directly on the NYT dataset, it still represents a significant increase over the not-augmented CNN-DM model and produces summaries that are quite readable. This technique could be used for low-resource domains and for problems with limited data availability.

## 7.7 ANALYSIS AND DISCUSSION

Method	R-1	R-2	R-L
LEAD-3	40.1	17.5	36.3
NEUSUM ( <i>Zhou et al., 2018</i> )	41.6	19.0	38.0
Top-3 sents (Cont. Select.)	40.7	<b>18.0</b>	37.0
Oracle Phrase-Selector	67.2	37.8	58.2
Content Selector	<b>42.0</b>	15.9	<b>37.3</b>

**Table 7.4:** Results of extractive approaches on the CNN-DM dataset. The first section shows sentence-extractive scores. The second section first shows an oracle score if the content selector selected all the correct words according to our matching heuristic. Finally, we show results when the Content Selector extracts all phrases above a selection probability threshold.

**EXTRACTIVE SUMMARY BY CONTENT SELECTION?** Given that the content selector is effective in conjunction with the abstractive model, it is interesting to know whether it has learned an effective extractive summarization system on its own. Table 7.4 shows experiments comparing content selection to extractive baselines. The LEAD-3 baseline is a commonly used baseline in news summarization that extracts the first three sentences from an article. Top-3 shows the performance when we extract the top three sentences by average copy probability from the selector. Interestingly, with this method, only 7.1% of the top three sentences are not within the first three, further reinforcing the strength of the LEAD-3 baseline. Our naive sentence-extractor performs slightly worse than the highest reported extractive score by *Zhou et al. (2018)* that is specifically trained to score combinations of sentences. The final entry shows the performance when all the words above a threshold are extracted such that the resulting summaries are approximately the length of reference summaries. The oracle score represents the results if our model had a perfect accuracy, and shows that the content selector, while yielding competitive results, has room for further improvements in future work.

This result shows that the model is quite effective at finding important words (ROUGE-1) but less effective at chaining them together (ROUGE-2). Similar to *Paulus et al. (2018)*, we find that the decrease in ROUGE-2 indicates a lack of fluency and grammaticality of the generated summaries. A typical example looks like this:

Data	%Novel	Verb	Noun	Adj
Reference	14.8	30.9	35.5	12.3
Vanilla S2S	6.6	14.5	19.7	5.1
Pointer-Generator	2.2	25.7	39.3	13.9
Bottom-Up Attention	0.5	53.3	24.8	6.5

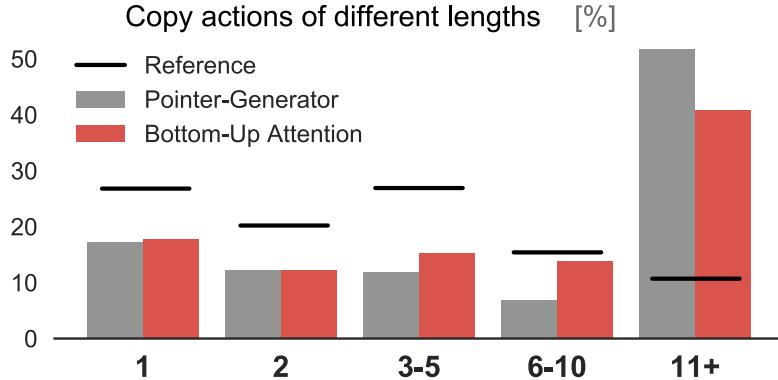
**Table 7.5:** %Novel shows the percentage of words in a summary that are not in the source document. The last three columns show the part-of-speech tag distribution of the novel words in generated summaries.

a man food his first hamburger wrongfully for 36 years. michael hanline, 69, was convicted of murder for the shooting of truck driver jt mcgarry in 1980 on judge charges.

This particular ungrammatical example has a ROUGE-1 of 29.3. This further highlights the benefit of the combined approach where bottom-up predictions are chained together fluently by the abstractive system. However, we also note that the abstractive system requires access to the full source document. Distillation experiments in which we tried to use the output of the content-selection as training-input to abstractive models showed a drastic decrease in model performance.

**ANALYSIS OF COPYING** While Pointer-Generator models have the ability to abstract in summary, the use of a copy mechanism causes the summaries to be mostly extractive. Table 7.5 shows that with copying the percentage of generated words that are not in the source document decreases from 6.6% to 2.2%, while reference summaries are much more abstractive with 14.8% novel words. Bottom-up attention leads to a further reduction to only a half percent. However, since generated summaries are typically not longer than 40-50 words, the difference between an abstractive system with and without bottom-up attention is less than one novel word per summary. This shows that the benefit of abstractive models has been less in their ability to produce better paraphrasing but more in the ability to create fluent summaries from a mostly extractive process.

Table 7.5 also shows the part-of-speech-tags of the novel generated words, and we can observe an interesting effect. Application of bottom-up attention leads to a sharp decrease in novel adjectives



**Figure 7.4:** For all copied words, we show the distribution over the length of copied phrases they are part of. The black lines indicate the reference summaries, and the bars the summaries with and without bottom-up attention.

and nouns, whereas the fraction of novel words that are verbs sharply increases. When looking at the novel verbs that are being generated, we notice a very high percentage of tense or number changes, indicated by variation of the word “say”, for example “said” or “says”, while novel nouns are mostly morphological variants of words in the source.

Figure 7.4 shows the length of the phrases that are being copied. While most copied phrases in the reference summaries are in groups of 1 to 5 words, the Pointer-Generator copies many very long sequences and full sentences of over 11 words. Since the content selection mask interrupts most long copy sequences, the model has to either generate the unselected words using only the generation probability or use a different word instead. While we observed both cases quite frequently in generated summaries, the fraction of very long copied phrases decreases. However, either with or without bottom-up attention, the distribution of the length of copied phrases is still quite different from the reference.

**INFERENCE PENALTY ANALYSIS** We next analyze the effect of the inference-time loss functions. Table 7.6 presents the marginal improvements over the simple Pointer-Generator when adding one penalty at a time. We observe that all three penalties improve all three scores, even when added on

Data	R-1	R-2	R-L
Pointer Generator	36.3	16.2	33.4
+ Length Penalty	38.0	16.8	35.0
+ Coverage Penalty	38.9	17.2	35.9
+ Trigram Repeat	39.1	17.4	36.1

**Table 7.6:** Results on CNN-DM when adding one inference penalty at a time.

top of the other two. This further indicates that the unmodified Pointer-Generator model has already learned an appropriate representation of the abstractive summarization problem, but is limited by its ineffective content selection and inference methods.

## 7.8 CONCLUSION

In this chapter, we presented a simple but accurate content selection model for summarization that identifies phrases within a document that are likely included in its summary. This mechanism follows the inherent structure of the summarization problem and simulates a similar reasoning process to that of humans. The content selection is thus understandable and meshes with the mental model that humans might have of the problem.

We showed that the content selector can be used in a bottom-up attention that restricts the ability of abstractive summarizers to copy words from the source. The combined bottom-up summarization system leads to improvements in ROUGE scores of over two points on both the CNN-DM and NYT corpora. A comparison to end-to-end trained methods showed that this particular problem cannot be easily solved with a single model, but instead requires fine-tuned inference restrictions. Finally, we showed that this technique, due to its data-efficiency, can be used to adjust a trained model with few data points, making it easy to transfer to a new domain.

Since the content selection provides the means to control the reasoning process of the model, we have the second essential building blocks of collaboration, the controllability. We will show in the next

chapter how we can put the building blocks together as part of a collaborative interface.

*They told me computers could only do arithmetic.*

Grace Hopper

# 8

## Collaborative Semantic Inference

One way to achieve collaboration is through the ability to have a discourse, where agents alternate between explaining their reasoning process and giving feedback on potential improvements. Through this iterative process, the agents should come to a mutually agreed upon solution to a problem, often one that is better than a single agent could have achieved alone.

This collaborative behavior within autonomous agents requires them to have the ability to (1) explain their reasoning process, and (2) update the reasoning based on received feedback. To accomplish

this goal, researchers from many disciplines argued that intelligent systems should be designed as team members (Grosz, 1996, Horvitz, 1999, Amershi et al., 2019, Heer, 2019, *inter alia*). In that case, the model output could enrich humans instead of replacing them. However, following this design process is not achievable with standard deep learning-based models.

To accomplish the goal, we thus need to design **interactive models** instead of autonomous models. To fulfill the two requirements, the interactions of a model must be easily understandable by a human end user. Therefore, we need to consider the interaction design when modeling a language generation problem. The designed interactions have to be reflected within the model design, which is an unsolved challenge for deep learning models.

To approach the problem, we propose a framework that can be applied to enable users to control predictive processes called *collaborative semantic inference* (CSI).<sup>1</sup> CSI describes a dialogue, alternating between model predictions presented in a visual form and user feedback on internal model reasoning. This process requires exposing the model’s internal process in a way that meshes with the *users mental model* of the problem and then empowering the user to influence this process. This approach is centered around the core design principles for visual analytics, which integrates visualization and analytics in a human-centered interface (Keim et al., 2008). Endert et al. (Endert et al., 2012) define semantic interactions as those which “enable analysts to spatially interact with [such] models directly within the visual metaphor using interactions that derive from their analytic process.” CSI describes *how* to connect these semantic interactions to the model inference process. The development of CSI methods and interfaces further requires a tight collaboration between the visual analytics, interaction design, and machine learning experts, which is a challenging, but promising, direction of research (Sacha et al., 2019, Endert et al., 2017, Stolper et al., 2014, Hohman et al., 2018).

As described before, deep neural networks do not expose their internal reasoning process. The CSI framework thus requires the development of model extensions that expose intermediate reasoning

---

<sup>1</sup>This chapter presents results from Gehrmann et al. (2019a).

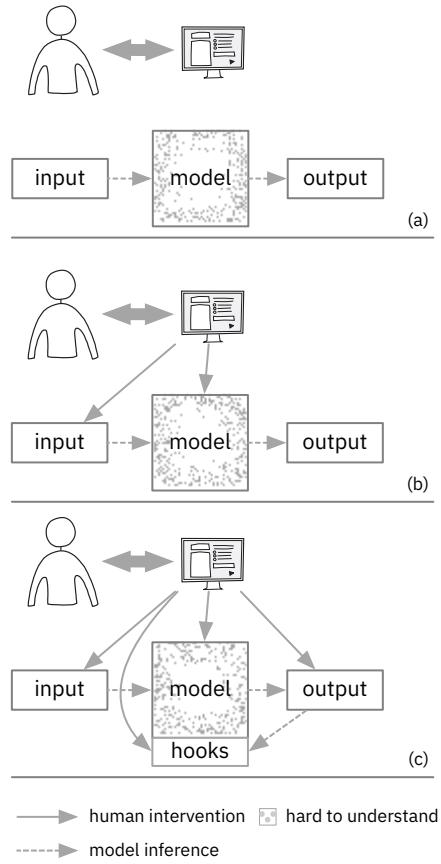
that can be associated with user-understandable choices. Our proposed solution to this problem is to incorporate discrete latent variables (Kim et al., 2018) into the model design. These variables act as “hooks” that can control the reasoning process and output of a model. The hooks enable what-if analyses by answering what internal choices would have led to a specific output. Crucially for CSI, the hooks also allow a user to infer the model’s reasoning process by seeing how a given output was selected. This visual analysis in the backward direction, from prediction to input, is typically not possible without model modifications.

To contextualize the multi-disciplinary co-design process and assist in developing collaborative tools, we extend the the visualization and interaction design space for neural network models we introduced in Chapter 3.2. Specifically, we extend definition of interpretability tools for model- and decision-understanding with the potential for model- and decision-*shaping*. The core difference here is that, through interaction, the model or decisions themselves can be shaped by human end users. The shaping-based techniques require an even tighter coupling between the model and the interface than interactive observation tools, and even change the model-design itself. We, therefore, add a third category *interactive collaboration*, which can be achieved through methods like CSI.

As proof of concept, we apply our design process to the use case of a document summarization system. When this task is handled by an automated system, the results almost always require heavy post-editing by humans. Our use-case presents a collaborative, deep learning-based, interface for this problem. This use-case also demonstrates the expanded design space of interactive visual interfaces for collaborative models.

## 8.1 INTERACTIVE COLLABORATION

We define collaboration in interactive interfaces for deep learning models as the ability of end-users or trainers with domain knowledge to present feedback to the model. The goal of collaboration is to



**Figure 8.1:** The categorization of interface types for collaboration with and interpretation of neural networks. As we move from passive observation (a) to interactive observation (b) and interactive collaboration (c), the interface requires a tighter integration with the model.

shape the model or its decisions, which means that either the decisions or the model itself update based on the human feedback. We call these *backward* interactions. Since each interaction direction needs to call a different shaping process within the model, the interface and model in interactive collaboration tools require a tight coupling. Only co-designing the model, visualizations, and interactions can achieve this tight integration.

Figure 8.1 illustrates the difference between interactive collaboration (c), passive observation, and interactive observation.

**MODEL-SHAPING** On the model-level, the feedback is expressed as user-provided labels which can be used to change the model parameters in an active learning setting (Holzinger, 2016, Cohn et al., 1996, Jiang et al., 2019). In the forward direction, model performance can be visualized, and new samples for the labeling process can be selected (Holzinger, 2016). The model parameters can be updated through backward interactions (Smilkov et al., 2017a, Kulesza et al., 2010).

**DECISION-SHAPING** Interactive collaboration for decision-shaping requires an interface in which the end-user can guide the model-internal reasoning process to generate a different output than the model would have reached on its own. Since interactive collaborative interfaces also retain the ability for forward interactions, the intervention enables an interplay between suggestions by the model and feedback by the user. Our proposed approach to developing CSI methods, which we describe in Section 8.2, presents one way to design such applications.

During forward interactions, the visualization shows what the model-internal reasoning process looks like for a specific input. During backward interactions, the end-user can modify the output and observe how the model-internal reasoning process would have looked to arrive at that specific output. The incorporation of these feedback mechanisms into visual analytics tools requires three essential components. First, the model needs to expose an interpretable hook along its internal reasoning chain, which should be transparent in derivation and understandable for non-experts (Lipton, 2018). Second, this interpretable hook needs to correspond to the mental model of the end-user. Most importantly, a collaborative tool needs to enable efficient interactions with the visual metaphor of the hook through semantic interaction (Endert et al., 2012).

The interpretable hooks of a model can act as explanations for rules of behavior that models learn. These explanations have been shown to improve model personalization (Bostandjiev et al., 2012) and explainability (Caruana et al., 2015, Kulesza et al., 2015). Conversely, failing to provide explanations can inhibit the development of mental models in end-users (Lim et al., 2009). However, explanations

should also not overwhelm an end-user, and many previous approaches thus choose to select less complex models (Lacave and Díez, 2002) or aim to reduce the feature space of a trained model (Craven and Shavlik, 1997; Yang and Pedersen, 1997).

CSI systems have models and end-users collaborate on the same output. This contrasts with previous work that often treats the model as a complementary assistant, for example recommending citations for a writer (Babaian et al., 2002). Moreover, CSI argues for a design approach to collaborative interfaces where the user retains agency over the exposed parts of the model’s reasoning process. Even in related approaches where the model and user both generate content, the users either do not have control over the model suggestions (Guzdial et al., 2017) or the model is replaced by uncontrollable crowdworkers (Bernstein et al., 2010). While previous work on interactive phrase-based machine translation showed promising results towards the goal of collaborative interfaces, the same techniques are not possible with deep learning-based approaches (Green et al., 2014). This lack of previous work can in part be attributed to deep learning methods having only recently reached the performance levels necessary for CSI-style interfaces.

## 8.2 REARCHITECTING MODELS TO ENABLE COLLABORATIVE SEMANTIC INFERENCE

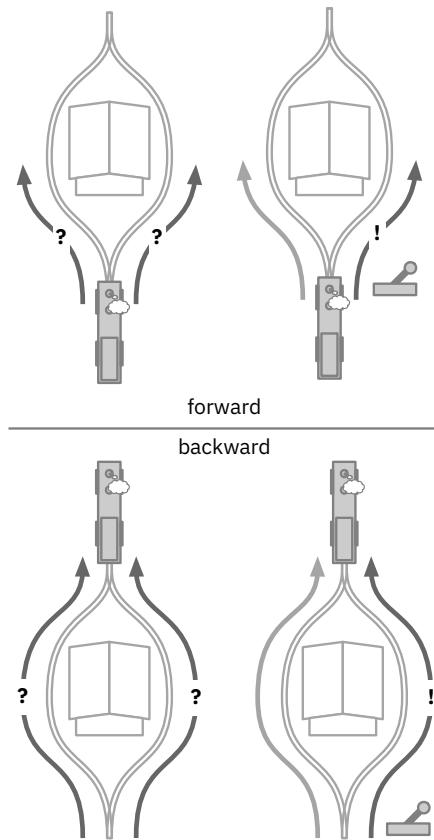
Interactive collaboration requires interpretable model hooks that enable semantic collaboration. From the machine learning side, these hooks can be implemented as discrete latent variables. During the prediction, or *inference*, process, the variables take on explicit values. Additionally, the variables must reflect an understandable aspect about the problem, that is being modeled, such that the explicit values are meaningful to a human user.

To illustrate this process, we consider a hook resembling a lever that directs a train toward a left or right track, as shown in Figure 8.2. A model is predicting where a train will end up. Without the hook, the model can predict the end position accurately, but it is not clear how it will get there

(top-left). Similarly, once the train reaches the top of the tracks, the model cannot explain how it got there (bottom-left). With the hook, however, the prediction explicitly exposes the lever decision. That means that the user can directly observe the position of the lever and thus knows whether the train will take the left or right track (top-right). Once the train is at the top of the intersection, the user can look at the position of the lever to find the path the train has taken (bottom-right). Most importantly, once the hook has been exposed, an end user can constrain or overwrite the decision-process in the model. If she wants the train to always take the right path, she could set the lever to the right position. This would not be possible without exposing this decision.

We next consider the summarization problem from Chapter 7. If we think of the content selection as the decision to use a word in a summary, we can make the summarization model controllable in the same way. Recall that summarization aims to generate a sequence of words  $y_1, \dots, y_T$  that is conditioned on an input  $\mathbf{x}$  using a deep model, where  $\mathbf{x}$  represents a long document and  $\mathbf{y}$  the summary. A forward-only deep sequence model defines a conditional distribution to predict one word at a time,  $p(y_{t+1} | \mathbf{y}_{1:t}, \mathbf{x})$  while considering all previously generated words. To formalize hooks for the collaborative approach, the deep sequence model is extended to expose intermediate terms as latent variables  $\mathbf{z}$ . In the summarization example, this could be the decision of which words in an input are considered important enough to be included in a summary. The architect defines  $p(y_{t+1} | \mathbf{y}_{1:t}, \mathbf{x}) = \sum_{\mathbf{z}} p(y_{t+1} | \mathbf{y}_{1:t}, \mathbf{x}, \mathbf{z}) \times p(\mathbf{z} | \mathbf{y}_{1:t}, \mathbf{x})$  for the latent variable  $\mathbf{z}$ . As a result, the model considers all possible values of  $\mathbf{z}$  to make a prediction. In the train-lever example, doing this enables us to run the inference process to compute the best lever position by looking at which of the two positions would lead to a better final position as judged by the model.

This approach splits the black-box into multiple parts: a *prediction network*,  $p(y_{t+1} | \mathbf{y}_{1:t}, \mathbf{x}, \mathbf{z})$ , that predicts the next word, and a *hook network*,  $p(\mathbf{z} | \mathbf{y}_{1:t}, \mathbf{x})$ , that predicts the value of the latent variables  $\mathbf{z}$ . Because this model is probabilistic, we can also perform posterior or *backward inference*. This gives  $p(\mathbf{z} | \mathbf{y}_{1:T}, \mathbf{x})$ , the distribution of  $\mathbf{z}$  after taking into account the entire output, for example



**Figure 8.2:** The advantage of latent variables is the transparent reasoning process. CSI requires model reasoning, illustrated by a lever that dictates which turn a train is taking. In the forward inference direction, we know in both cases where the train will end up, but only the lever allows us to know what path it will take. Similarly, in the backward inference direction, we know where the train originated, but only the lever shows the track it took to get there.

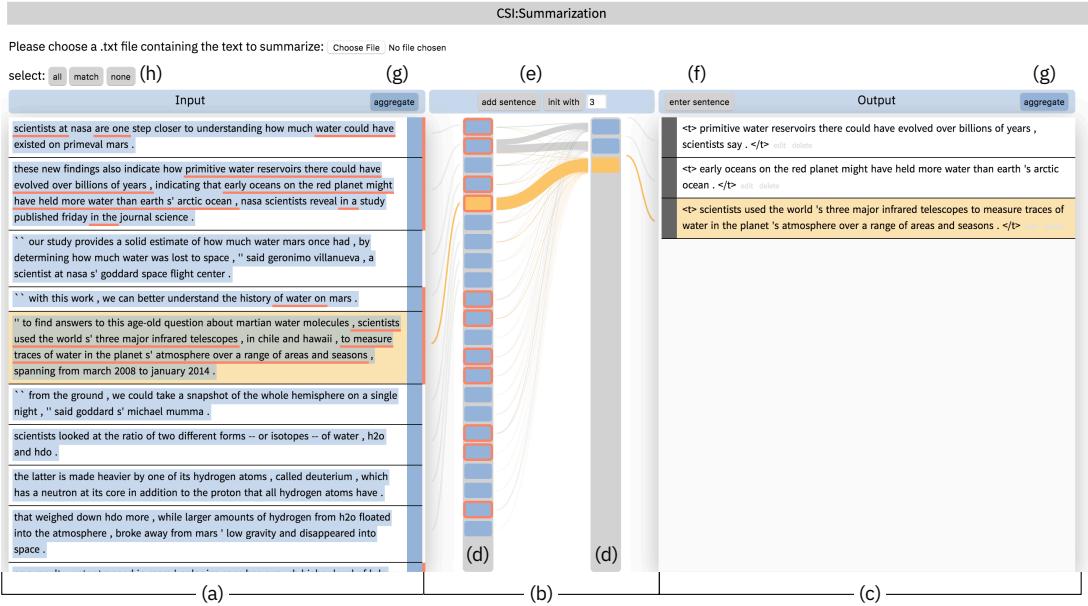
the summary.

Another example of a collaborative interface could be for semantic image synthesis ([Chen and Koltun, 2017](#), [Wang et al., 2018b](#), [Park et al., 2019](#)). In this application, a user-defined input  $\mathbf{x}$  describes high-level features, for example, the location of grass and sky, and a neural network generates the corresponding image. For this case,  $\mathbf{y}$  is an image and not a sequence of words. Current approaches do not allow iterative refinement through interaction with an image and are limited to changing the input and generating a completely new output. A collaborative approach to the same problem is

GANPaint (Bau et al., 2019), which uses a hook network to associate parts of the latent space of an image-generating model with semantic features and exposes a modification interface to the user.

In a real-world example of a hook-network, Google Translate recently introduced a semi-collaborative approach to preventing gender-discrimination in translation systems. By treating gender as a hook, they can present all possible options for gendered pronouns in a translations when the gender in a source language is ambiguous. This approach allows a user to pick the translation they want. At this time, the approach has been implemented for Turkish, which is a language without grammatically gendered nouns and pronouns.

The model hooks represent ways in which users can constrain and overwrite interpretable decisions in otherwise end-to-end black boxes. They are extensions of otherwise well-performing models to expose the latent variable. Hooks must be co-designed by experts in interaction design, visualization, and machine learning. They decide together on the model hooks, desired interactions, and the associated visual encodings. While some guidelines have been developed for interaction design for machine learning (Fails and Olsen Jr, 2003, Stumpf et al., 2009, Amershi et al., 2014, Yang, 2018), they focus on cases where the model performs complementary tasks to the user, or where the user interacts with black-box models. In contrast, CSI enables the study of interaction design for models that approach the same task as the user. One important question that requires further study is how many hooks are actually useful to a user. As more latent variables are designed and incorporated into a model, the training process becomes increasingly more challenging and model performance might degrade. Moreover, the increase in potential user interactions with additional hooks might overwhelm users. As a consequence, we focus on a model with a single hook throughout our use case and show how even a single hook can enable many powerful interactions.



**Figure 8.3:** Overview of the CSI:Summarization visual interface. (a) shows the input and overlays the currently selected content selection (blue) and the current content of the summary (red). (b) shows a connection between the input and the current summary through the attention, which shows explicitly where words in the summary came from. (c) shows the current summary, (d) shows proxy elements for both input and output groups. This enables an overview of a document, even when the text does not fit on one page. (e) allows the user to request suggestions from the model, (f) enters the edit mode and adds a new sentence to the summary. (g) toggles whether the text should be aggregated into sentences. (h) provides quick selections for the content selection (blue) by being able to match the red highlights, or (de-)select everything.

### 8.3 USE CASE: A COLLABORATIVE SUMMARIZATION MODEL

We demonstrate an application of the CSI framework to the use case of text summarization. Text summarization systems aim to generate summaries in natural language that concisely represent information in a longer text. This problem has been targeted by the deep learning community using an approach known as sequence-to-sequence models with attention (Bahdanau et al., 2015, Sutskever et al., 2014). These models have three main architectural components: (1) an encoder that reads the input and represents each input word as a vector, (2) a decoder that takes into consideration all previously generated words and aims to predict the next one, and (3) an attention mechanism that represents an

alignment between the current decoding step and the inputs. In summarization, the attention can be loosely interpreted as the current focus of the generation and can be visualized for each generated word (Strobelt et al., 2019).

Imagine an end-user called Anna using an interface powered by a summarization model. Current deep learning models act in a forward-only manner, as described above; therefore the design space is limited to an interactive observation interface. This interface allows Anna to paste an input text and have the model infer an output summary. If Anna does not like the output summary, she can edit the suggestion to her liking, but it is not possible to reuse the model to check her changes. Moreover, if the model produced a bad or wrong output, Anna would have to write the entire summary from scratch.

Applying the CSI framework and extending a well-performing summarization model with the previously defined hooks can address these issues. By tying the user’s interactions to understandable reactions from the model, we can achieve three types of previously not possible interactions. The collaborative interface (1) **guides the model** towards important content, (2) **enables a dialogue** between human-generated and machine-generated output, and (3) allows a user to **review the decisions** the model would have made to generate a specific output, i.e., what parts of an input text the model chose to summarize.

These changes require designing a semantic model hook that can describe the content that a model considers for a summary. We, therefore, formalize content selection as a hook for each word in a document. By exposing the hook within the interface, we can describe the semantic interaction as the user-decision what content of a document is relevant for its summary.

## 8.4 DETAILS ON THE SUMMARIZATION MODEL HOOKS

We base the summarization model on the bottom-up model described in Chapter 7. As with that model, we use a sequence-to-sequence model with an attention distribution  $p(\mathbf{a}_t|\mathbf{x}, \mathbf{y}_{1:t})$  for each decoding (generation) step  $t$  over all the source words in  $\mathbf{x}$ , which is calculated as part of the neural network. Our model additionally uses the same copy mechanism to enable the model to either generate a new word or copy a word from the source document during the generation. The copy-attention is directly interpretable since a high value means that the model is actually copying a specific word from the input.

As we demonstrated, this approach yields high performance on automatic metrics, but the end-to-end approach with decisions per generated word is disconnected from human summarization approaches. Specifically, we identified that the model does not follow the human-like approach of first deciding what content within a document is important, and then trying to paraphrase it ([Jing and McKeown, 1999](#)). However, the bottom-up mechanism we introduced can be used as a controllable hook within the model. The *hook network* predicts the probabilities  $p(\mathbf{t}|\mathbf{x})$  to decide what content is important. The *prediction network* that generates the summary  $p(y_{t+1}|\mathbf{x}, \mathbf{y}_{1:t}, \mathbf{t})$  uses these probabilities to avoid copying unimportant content. Since the model explicitly reasons over content-importance through the hook network, we can achieve semantic interactions by letting users define a prior on  $p(\mathbf{t}|\mathbf{x})$ . When user deselects a sentence from the input, we set the prior  $p(\mathbf{t})$  for all its words to 0, which means that the hook network can no longer identify the words as important which means that it is prevented from copying deselected words.

The last step towards the fully integrated CSI:Summarization is *backward inference*, i.e. the identification of what content a summary actually used, or  $p(\mathbf{t}|\mathbf{x}, \mathbf{y})$ . The backwards model is a separate model we specifically developed for the interface. It uses a contextualized representation of words in both input and summary that represents them as vectors of size  $D_{hid}$  ([Devlin et al., 2019](#)), denoted as

$\mathbf{x}_s$  and  $\mathbf{y}_t$ . Given the representation for a word  $x_s$ , the model computes an attention over  $\mathbf{y}$ , such that each summary word  $y_t$  is assigned an attention weight  $a_{s,t}$ . We use these weights to derive a context for the word  $x_s$  which we denote  $\mathbf{c}_s$ , by computing

$$\mathbf{c}_s = \sum_{t=1}^T a_{s,t} \times \mathbf{y}_t.$$

To arrive at a probability that  $x_s$  was used in the output, we apply a simple MLP,

$$p(t_s|\mathbf{x}, \mathbf{y}) = \sigma(\mathbf{W}^2 \tanh(\mathbf{W}^1[\mathbf{x}_s, \mathbf{c}_s] + \mathbf{b}^1) + \mathbf{b}^2),$$

where  $b^1, b^2$  are trainable bias terms and  $\mathbf{W}^1 \in \mathbb{R}^{D_{hid} \times 2D_{hid}}$  and  $\mathbf{W}^2 \in \mathbb{R}^{1 \times D_{hid}}$  trainable parameters. Since this model is independent of the forward model, it can analyze arbitrary summaries, even those that are written by the end user, as we show throughout the use case of Anna generating a summary by using our CSI interface to collaborate semantically with the model.

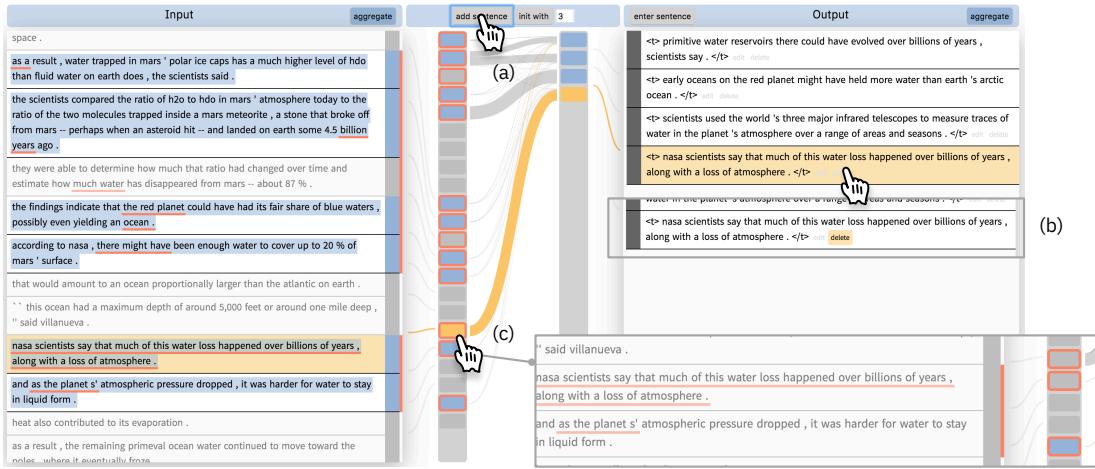
#### 8.4.1 COLLABORATIVE SUMMARIZATION: ANNA’s STORY

Anna intends to collaboratively write a summary of an article describing how scientists found water on Mars<sup>2</sup>. She begins by reading the article to assess what information is relevant and should be part of the summary. Then to begin the interaction, she selects the entire input text, shown by the blue highlights in Figure 8.3a, letting the model know it is free to summarize any relevant part of the document.

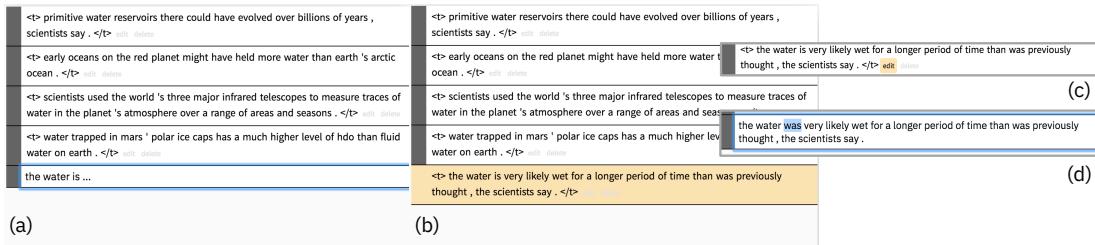
She starts the collaborative writing process by requesting that the model suggest three initial sentences (Figure 8.3c). This triggers a forward inference of the model and a visual update that presents the suggestions to Anna. At the same time, the system computes a backward inference to show which part of the input has been summarized. Visually, the input text may be longer than the browser window, so we introduce a proxy element for each sentence in input and output (Figure 8.3d). The words

---

<sup>2</sup>The article can be found at <https://www.cnn.com/2015/03/06/us/mars-ocean-water-study>



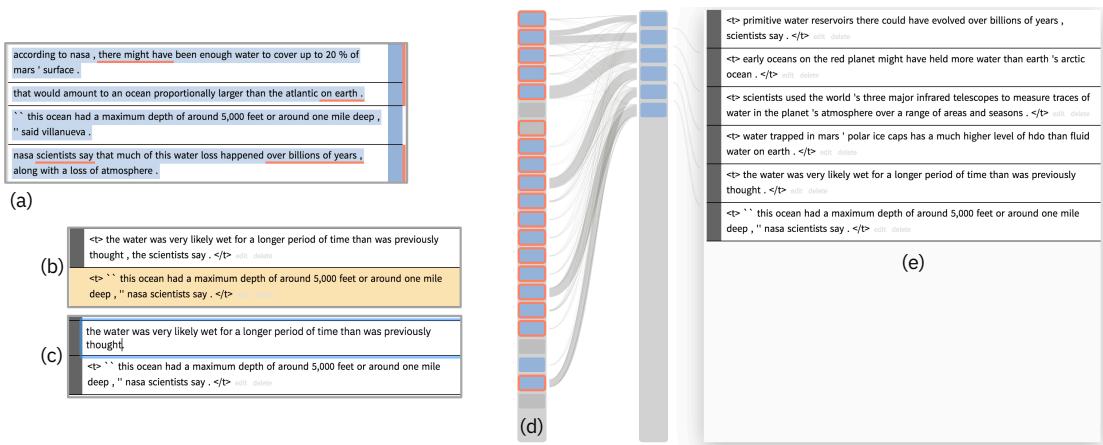
**Figure 8.4:** After selecting the content shown on the left, Anna requests the model to generate a fourth sentence (a). She does not like the suggestion and deletes it (b). To influence the model to generate about other topics in the input, she deselects the sentences that caused the suggestion (c).



**Figure 8.5:** Anna wants to generate a sentence about the water on Mars and starts typing “The water is ...” (a). This initiates the model to finish the sentence for her (b). To correct a minor mistake in the generated sentence, Anna activates the edit mode (c) and replaces the wrong word (d).

in the input that are related to the summary are presented with a red underline and the proxies of the sentences with at least one related word have a red border (Figure 8.3a,b). The interface also visualizes the model “attention,” which shows which covered words were selected by the model at what step during the generation. The attention is visualized using grey ribbons that are aggregated across each sentence and connect the proxy elements. If Anna hovers over one of the sentences or proxies, the interface highlights the relevant connections in yellow (Figure 8.3b).

Through these visual interactions with the output summary, Anna observes that the second input



**Figure 8.6:** Anna selects a sentence that is currently not covered by the summary, indicated by the lack of a red border on the right (a). She generates a new sentence (b) and corrects the repeated phrase “nasa scientists say” (c). With the finished draft of her summary, she can now evaluate the coverage of the input document (d) and the final summary (e).

sentence (“*scientists at nasa are one step closer to understanding how much water could have existed on primeval mars. these new findings also indicate how primitive water reservoirs there could have evolved over billions of years, indicating that early oceans on the red planet might have held more water than earth’s arctic ocean, nasa scientists reveal [...]”*”) splits into two different summary sentences (“*primitive water reservoirs there could have evolved over billions of years, scientists say.*” and “*early oceans on the red planet might have held more water than earth’s arctic ocean.*”). It is common for summarization models to compress, merge, and split input sentences, but the user would not be aware of where the inputs for each summary sentence originate. By exposing the internal model decisions in a CSI interface, a user can immediately discover how the input connects to the output.

From this suggested summary, Anna determines that the input focus of the system was correct, but that output text should elaborate on these sentences in more detail. She communicates this by first constraining the model to the currently focused region. She can match the content selection (blue highlights) with the result of the backward inference (red underlines) by clicking “match” in Figure 8.3h. With these constraints, she triggers the generation of an additional output sentence (“*nasa*

*scientists say that much of this water loss happened over billions of years, along with a loss of atmosphere.”*) (Figure 8.4a). This suggestion is the result of a forward inference with her additional constraint on the model hook.

Unfortunately, Anna is dissatisfied with the new sentence, so she removes it from the summary (Figure 8.4b). To prevent the model from suggesting the same sentence again, she consults the backward inference and deselects the input sentence that had the highest influence on its generation by clicking on its proxy element (Figure 8.4c). With this updated constraint, Anna generates another sentence (“*water trapped in mars’ polar ice caps has a much higher level of hdo than fluid water on earth.*”) that better captures her goals. (Figure 8.5a).

Anna would next like to include more details in the summary, particularly about water found on Mars. The system allows her to intervene in the output text directly. She starts writing “the water is” and then adds an ellipse (...) that triggers sentence completion by the model (Figure 8.5b,c). The resulting sentence (“*the water is very likely wet for a longer period of time than was previously thought, the scientists say.*”) is acceptable to her, but she now spots an error with the verb (“is”) which should be in the past tense. She quickly corrects this output (Figure 8.5d,e), which invokes a backward inference to the input document. By updating the red highlights in the input, the interface provides her with information about what content was selected was used to create this improved sentence. These interactions help her create a mental picture of the model behavior.

Finally, Anna would like the model to help her generate a sentence about a region of the input that is currently not included in the summary. She selects a previously unused sentence in the input (“*this ocean had a maximum depth of around 5,000 feet or around one mile deep, said villanueva.*”) (Figure 8.6a), requests another forward inference, and approves of the resulting suggestion (“*this ocean had a maximum depth of around 5,000 feet or around one mile deep, nasa scientists say.*”). However, she dislikes the repetition of “scientist say” (Figure 8.6b). After entering the edit mode on the output side, she removes one of the repeated phrases (Figure 8.6c). Upon leaving the edit mode, the interface

automatically triggers another backward inference that updates which parts of the inputs are covered. Anna uses this information to evaluate how much of the document is covered by her summary. By looking at the computed coverage (Figure 8.6d), she can observe which sentences are covered and analyze how many of the proxies have a red border. She decides that, by this metric, her six sentences (Figure 8.6e) are an appropriate representation of the content in the original text.

#### 8.4.2 VISUAL AND INTERACTION DESIGN

We designed the text summarization prototype (*CSI: Summarization*) such that text occupies the majority of screen estate as the central carrier of information for the task. Two central panels (Figure 8.3a,c) represent input text and output text. Each text box represents words that are aggregated into sentences. Text highlights in the input show information about the model hooks and relations between input and output. Neutral gray colors are used on the output side to clearly distinguish them from the blue colors that represent selections on the input (Figure 8.3c).

The input and output text are connected by a bi-partite graph that indicates model attention (Figure 8.3b), which expands on previous work on visualizing and normalizing attention (Strobelt et al., 2018a, Lin et al., 2018). Due to the length of source documents, displaying the entire graph is not feasible or informative. Therefore, we use two design elements to enable users to observe the full graph in a de-cluttered view: aggregations and proxies. First, we allow the aggregation of words into meaningful word groups, e.g., sentences, that can be dissolved on demand (Figure 8.7d) if this level of detail is required. Aggregating words implicitly requires the aggregation of the attention which simplifies the graph. Secondly, we represent sentences by a vertical arrangement of boxes that are space-filling in height and which act as proxies for the full sentences. In that way, all sentences are always visible by their proxy, even when they are outside the display area. These proxies mirror selections and highlights of their related text boxes.

*CSI: Summarization* offers a range of user interactions. As a general principle, buttons trigger for-

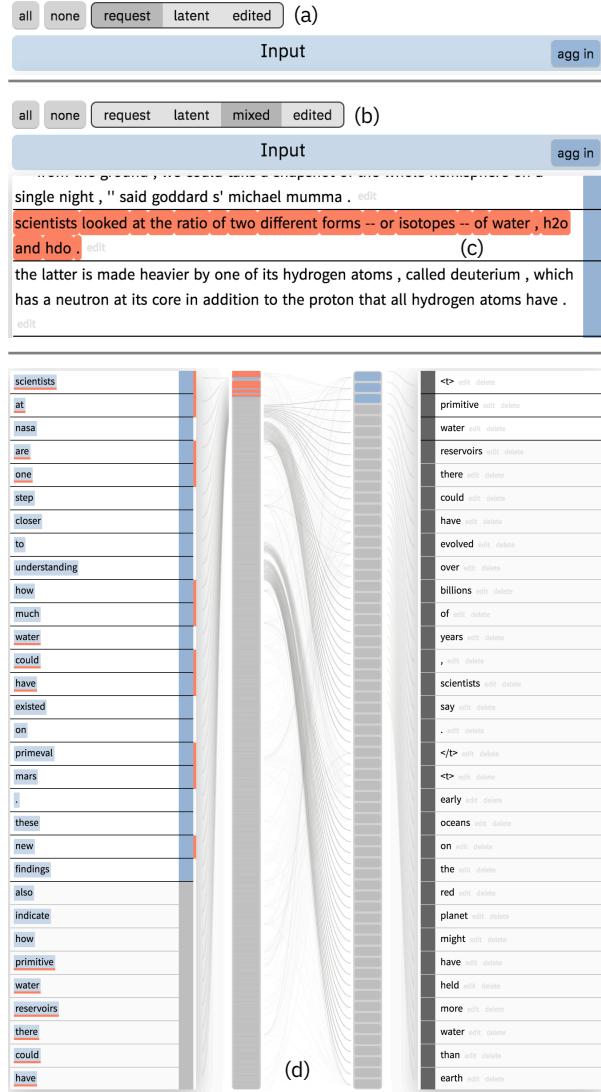
ward (left to right) actions (Figure 8.3e) because a forward inference can change the output summary itself. Unintended changes to the output could confuse the user. To let users explicitly request updates instead of automatically intervening is inspired by similar mixed-initiative writing assistants ([Babaian et al., 2002](#)), where researchers found that this type of interaction is seen as least intrusive. All backward inferences are automatically triggered after exiting the edit mode by hitting enter. Since backward interactions do not change the content of the summary, the automatic invocation does not lead to accidentally overwriting important information. Users can define which sentences or words to consider for generating the next sentences by selecting them (blue color). Clicking on the bar on the right of an aggregation group selects or deselects the entire group. The same action is triggered by clicking on the proxy element of the corresponding sentence (Figure 8.3d).

The interface additionally provides three selection templates (Figure 8.3h) for convenience: select all sentences, select no sentence, or select only those sentences that match the selection from the backward step (match red and blue). For the forward pass, the selection can be used to either initialize a new summary with a user-selected number of sentences (`init with`) or to add a sentence to the existing summary (`add sentence`) (Figure 8.3e). On the output side, sentences can be deleted or edited by clicking the `edit` and `delete` buttons at the end of each sentence.

#### 8.4.3 DESIGN ITERATIONS

During the creation of the prototype, we explored multiple designs for model hooks, visualizations, interactions, and their integration. Overall, we found that CSI systems are more difficult to design because of the, sometimes competing, interactions between all these elements. We want to highlight one example for each of the elements.

On the model side, our initial backward inference had good accuracy but did not reveal useful information within the interface. Only after re-allocating efforts to develop a different model with a much higher performance did the backward model match human intuition. Since model hooks

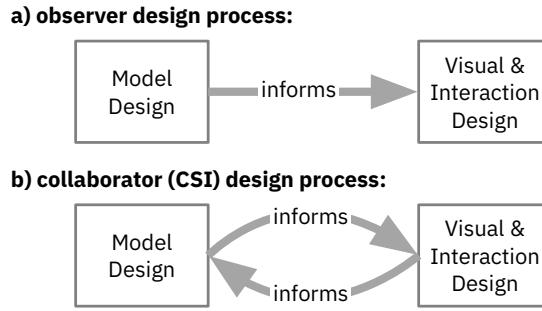


**Figure 8.7:** Visual design iterations for CSI:Summary. (a) shows radio buttons for selecting a specific selection (content selection vs. backward model). In (b), we introduced a mixed mode that showed the user content selection in the final blue color and the result from the backward model with a red highlight (c). Underlines later replaced the dominant highlight. (d) shows an example of the complexity of the attention graph without any aggregation.

complicate the machine learning models, it also complicates their training process. This can lead to issues where the model decisions are not useful to human end-users which can result in a decreased overall efficiency.

On the visualization side, we explored multiple different designs for the input selections presented in the interface. The selections we currently show are (1) the selection that was used for the most recent forward step, (2) the selection that was returned from the most recent backward step, and (3) the sentences that the user is selecting for the next forward pass. In a first iteration, we aimed to show all of them in separate views (Figure 8.7a), and additionally have a view that highlights their intersection (Figure 8.7b,c). Pilot studies revealed that only the combined view was useful to users to avoid having to switch forth and back. We, therefore, replaced the different views by the current more natural and coherent use of red and blue highlights within the same view. This iteration illustrates the challenge that the information from the model requires an appropriate visual representation within an interface. the developers of CSI interfaces need to consider the necessary abstraction between model internals and end-users intuitive understanding.

On the interaction side, we found that requesting that the model generate words without a constraint on the minimum or the maximum number of sentences often led to output that was unreasonable to users by repeating itself or being ungrammatical. The model architects on our team pointed out that the training data for summarization models rarely contains examples where the summary is longer than three sentences. Forcing a model to generate longer summaries than it was trained to generate led to degradation in output quality. We also found that users had more control over the content of a summary if they iteratively built up a summary from a short initial suggestion instead of having the model suggest a lengthy summary and letting the user change it afterward. Our current modes combine these findings by designing the interaction after discussions with model architects and visualization experts. The first interaction initially generates a small number of sentences. This leads to better model output and also lets users explore the output space more effectively. Similarly, adding one new sentence at a time by incorporating previous sentences as prefix context and allowing users to select the content enables users to quickly generate and review new content.



**Figure 8.8:** CSI interfaces require a design process that spans machine learning and visual interaction design. The feed-back loop is used to produce interpretable semantic representations that inform the reasoning procedure of the model while providing useful visual interaction.

## 8.5 TOWARDS A CO-DESIGN PROCESS FOR CSI SYSTEMS

During the implementation of the *CSI:Summarization* prototype we developed an understanding of how an integrated design process for CSI systems could look and also experienced its limitations. We discuss our insights as learned lessons.

**Prioritize collaborative output.** CSI systems enable joint production by model and end-user together. The resulting output must be the central element for developing visualization and interaction ideas. It is essential to evaluate if a *CSI approach is beneficial* for a given task, e.g., a face recognition model used to unlock a cell phone does not benefit from the CSI approach as no shared output is produced. Since *CSI methods are decision-shaping*, they require human oversight and interventions and are thus not suited for processing massive data. Moreover, since CSI interfaces are targeted at end-users, the visualization can be domain-specific but should abstract model-internals in an intuitive way. Finally, *no agent should dominate over the other* to allow model-human collaboration, which should be reflected in the visualization and interaction design – e.g., allowing equal easy access to triggers for human input and model suggestions.

**CO-DESIGN REQUIRES CONTINUOUS EVALUATION.** In the development of the summarization system, there was a *constant negotiation between visualization requirements and model capabilities*. This process led to iteration on the question: “does this visual encoding help the end-user collaborating with the model?” and “which additional model behavior do we need to help to encode relevant information?”. Figure 8.8 illustrates how this continuous co-design forms a bilateral relationship between model design and visualization design. CSI systems, like most visual analytics systems, can help to *reveal model problems* immediately. If, for example, a specific model hook is performing so poorly that it cannot facilitate the user’s mental model, it will be immediately revealed. On the other hand, requirements for a *visualization might over-constrain a model* such, that it breaks. E.g., creating a system for writing poetry that suggests lines of text that rhyme, even for human entered text, might over-constrain the model. CSI systems aim to find the middle ground between the ideal user experience and what is possible with the underlying ML model.

**CSI MAY BE A WORTHWHILE INVESTMENT.** Since CSI is centered around a single abstraction that reflects the mental model that an end-user has of a problem, we propose that *machine learning experts need to study possible interactions*. There is currently a limited understanding of the space of easily trainable hooks and interaction strategies. Since machine learning techniques do not natively consider bidirectional interactions with end-users, the visualization, interaction design, and machine learning experts need to teach each other about desired interactions and the limits of deep models. Deep learning models with hooks thus lead to an increased development complexity for both machine learning and visualization experts. However, during the development of the summarization use case, we also experienced that *CSI has a learning curve*. While CSI systems are individualized to a problem and thus one-of-a-kind systems, most of the techniques are transferable and a research or product team can apply the insights gained from one CSI project to the next one. Moreover, the long-term benefits of the increased control over models can justify the additional development complexity, especially

considering that many applications in industry are used for many years. As shown in our summarization example, the CSI methods can even lead to more structure and subsequent improvements in outcomes.

## 8.6 CONCLUSIONS

In this chapter, we introduced a framework for collaborative semantic inference, which describes a design process for collaborative interactions between deep learning models and end-users. Interfaces designed within this framework tightly couple the visual interface with model reasoning. We applied CSI to develop a collaborative system for document summarization that demonstrates that CSI systems can achieve powerful interactions within an interface powered by a neural model.

While previous studies have shown that explainability methods can mediate in an agency-efficiency trade-off (Lai and Tan, 2019, Yin et al., 2019), none of them demonstrate a way for users to retain agency while gaining the efficiency benefits of models interaction. We believe this is due to the difficulty of engaging the user in the prediction process of a black-box model. We address this problem by designing semantic interactions as part of the model itself. While CSI does not solve problems with biased data and models or the lack of interpretability of models, it aims to expose important model decisions and facilitate collaboration between an end-user and the model to take these decisions. Further developments in interpretability research could be used in conjunction with CSI for a better overall model understanding.

The CSI framework significantly expands the interaction design space over conventional interaction strategies for many deep learning models. CSI-style approaches have the potential for application especially in scientific or safety-critical applications where explainable AI may become mandatory. Moreover, since CSI treats the model as a team member, another area of particular interest are creative applications, where models can assist users in creating stories (Fan et al., 2018), chord progres-

sions (Huang et al., 2019), or even recipes (Kiddon et al., 2016). Future work might also investigate how similar principles could support cases where more than a single end-user and a single model aim to collaborate.

Finally, it is crucial to develop ways to systematically evaluate collaborative interfaces and to investigate the implications of designing algorithmic interactions with humans (Wilks, 2010, Williams, 2018). While an interpretability-first approach could assist in highlighting fairness and bias issues in data or models (Hughes et al., 2018, Holstein et al., 2019), it could also introduce unwanted biases by guiding the user towards what the model has learned (Arnold et al., 2018). It is thus insufficient to limit the evaluation of a system to measures of efficiency and accuracy. Future work needs to address these shortcomings by developing nuanced evaluation strategies that can detect undesired biases and their effect on end-users.

We provide a demo, the code, and the required models for CSI:Summarization at [www.c-s-i.ai](http://www.c-s-i.ai).

# 9

## Discussion and Conclusion

Throughout this dissertation, we have presented approaches that aim to improve the generation of natural language by computer systems that deploy neural network technology. We have discussed approaches for the interpretation of the models. Finally, we have developed a way to collaborate with these models. However, as neural architectures become more powerful, we also need to consider the implications of these models. Two important questions arise related to the evaluation of generated text and the prevention of abuse of the models.

## 9.1 THE EVALUATION OF TEXT-GENERATING MODELS

In supervised machine learning tasks, a generated text is usually compared to one or multiple human-written reference texts. Evaluation metrics are based on the lexical word-overlap between the generated text and the reference. Some tasks like machine translation use precision-based metrics like BLEU ([Papineni et al., 2002](#)), and some tasks like summarization use recall-based metrics like ROUGE ([Lin, 2004](#)). The goal is to capture the performance of a model with a single number. Recall for example, that the summarization model presented in Chapter 7 led to a consistent 2-point improvement in ROUGE score. While this improvement in ROUGE score is a strong signal, there are two issues we address below: (1) It does not work in all types of writing tasks. (2) The metrics are flawed.

Many NLG tasks do not have a single correct answer. Consider story-generation. Stories are supposed to be engaging and to capture the reader’s undivided attention. They need to have consistent characters and a consistent timeline of events. They are not, however, supposed to have a significant overlap with what some writer has decided is the “correct” story for a given set of constraints. For that reason, it is not possible to apply automatic metrics to tasks like story-generation.

A similar example is that of abstractive summarization. The models we discussed in Chapter 7 are trained on news summarization tasks which are inherently extractive. News articles are written in a manner that presents a compressed version of the events at the beginning of the article. That means that there is a high overlap between the summary and the article. However, other summarization tasks, for example, for the generation of scientific abstracts ([Cohan et al., 2018](#)) or the summary of human-written social media posts ([Völske et al., 2017](#)), are more open-ended.

The second issue is that automatic evaluation metrics are often flawed or may be used in unintended ways. For example, ROUGE was initially used to compare length-constrained summaries, where each summary could have a maximum length of 75 bytes, which is approximately 14 words ([Over et al., 2007](#)). This restriction no longer exists. However, since ROUGE is a recall-based metric,

it strongly encourages capturing as many phrases from the reference summary as possible. Consequently, ROUGE gives an advantage to systems that produce longer summaries, which makes it impossible to compare different systems if they do not produce summaries of the same length (Sun et al., 2019). Some metrics have been proposed that aim to evaluate generated text with references and do not suffer from such drawbacks. For example, Zhang et al. (2019b) use the similarity in contextualized representations of reference and generation instead of computing lexical matches. Eyal et al. (2019) evaluate whether a question-answering model can identify entities in a reference summary from the information in the generated summary. However, these metrics still assume that a reference exists that exhibits all the desired properties of the generated text. The automatic evaluation of generated text thus remains an unsolved challenge.

An alternative approach for the evaluation of generated text is to conduct human-subjects studies. We identified two different strategies that can be used for the evaluation of generated text:

**FOCUSED USER STUDIES** When aiming to capture properties of text with automatic metrics, we often overlook how the generated text affects its readers ability to accomplish a task. Consider the journalist Anna who aims to summarize her article. If an assistive system is used to generate the first version of her abstract, this suggestion might affect her view of the article. For example, if the suggestion has a more negative sentiment than she intended, it might subconsciously bias her. Similar effects have been found for assistive keyboards, which can affect the sentiment and length of texts that users produce (Arnold et al., 2018).

One particularly exciting direction is measuring the effect that the text has on end users. For example, in previous work, we aimed to improve the understanding of long documents through the generation of section titles for each paragraph (Gehrmann et al., 2019b). A human-subjects study was conducted to compare not showing titles, human-generated titles, and automatically generated titles. For each subject, we further varied the difficulty level of the documents and asked them to

complete three different tasks related to document understanding: finding information, memorizing information, and summarizing the document. Through an analysis of the results, we found that there were only small effects on the accuracy of the finding and memorizing tasks, but significant effects on the time spent on these tasks.

Moreover, the summaries subjects generated when shown titles, were generally longer and more detailed, although the writing took less time. We further identified a profound difference between human- and machine-written titles. While the generated titles reduced the time for the information-related tasks more, the human-written titles had a more substantial effect on the summarization tasks. We hypothesize that the different effects stem from the different levels of abstractiveness of the summaries. The human summaries provide high-level overviews over the content of the paragraph, whereas our generation process compresses a single sentence and thus focuses on the content rather than the story. Investigations like this are only possible through user studies. However, due to the monetary cost and the difficulty of designing focused user studies, it is often challenging to get insights across a broad sample of the intended user group. In this case, long-term user studies can be beneficial.

**LONG-TERM USER STUDIES** An alternative way to capture feedback for a collaborative tool is to release an early version to a select group of testers or to release the tool to the general population. There are two conditions to consider for this evaluation path. First, it has to be ensured that the tool is well explained and documented. Often, users can find unintended ways to use a tool or misunderstand crucial aspects. While this can have beneficial consequences (see Chapter 5), it can also cause harm. The release can gather media attention which can lead to a misrepresentation of AI to the public, if not adequately explained. Other tools could be abused directly, for example, interactive generation tools for fake news articles (Zellers et al., 2019). It is thus crucial to verify the tool with a smaller group of testers of the prototype.

However, if a tool gathers enough interest, long-term user studies are a useful way to evaluate its

efficacy. The release should be accompanied by many possible opportunities to provide feedback, for example, through email, online discussions, social media, and GitHub issues. This feedback, in conjunction with an evaluation of the server logs (in accordance with privacy laws), can be used to substantially improve the tool. A disadvantage of this strategy is that the feedback is often undirected and noisy. Given the broad user group for a typically highly specialized tool, it is impossible to ask focused questions. In this case, a smaller user study might be more appropriate.

## 9.2 THE ETHICAL PERMISSIBILITY OF TEXT-GENERATING MODELS

As mentioned above, the abuse of collaborative interfaces becomes more likely as the text-generating models become more powerful (Brundage et al., 2018). In fact, research on scaling models to an unprecedented scale sparked a discussion around whether these large models should even be available to anyone (Radford et al., 2019). The underlying argument is whether powerful text-generating models are inherently harmful. Generated text from unconditionally trained language models has reached a point at which it is nearly indistinguishable from human-written text. While it is not currently possible to direct the content of the generated text, the controllability methods are rapidly advancing. Models like Grover (Zellers et al., 2019) and CTRL (Keskar et al., 2019) can generate text in a specific style and content by conditioning on certain control tokens. Human-like generated text has substantial implications for how online reviews, news, or opinions should be perceived. Moreover, it also has implications for the broader research community. Since NLG models are technologies with both beneficial and malicious applications, we need to consider how to prevent the malicious use of the technology.

Since these developments are recent, there are no common ethical guidelines that direct NLG research. In a survey, Fort and Couillault (2016) found that, while over 52.5% of researchers feel responsible for the use of the methods they develop, they point out that ethics is rarely included in

conferences and that the implications on society are rarely discussed in research papers. Over 91% of respondents in the survey by Fort and Couillaud believe that the public is insufficiently informed about the limits and possibilities of the tools we create. NLG researchers thus must develop these guidelines and inform the public about the possibilities of generated text. One complicating factor toward the ethical behavior of neural network-based systems is the trade-off between accuracy, robustness, and transparency (Thielges et al., 2016). For example, the increasing parameter sizes of language models led to more natural text but made the models less interpretable. We argue that the development of inherently interpretable neural architectures, as suggested in Chapter 8, is crucial to address and potentially side-step the trade-off. This statement is further corroborated by Wilks (2010) who argues that because algorithms will never be perfect, they should work together with humans instead of replacing them. As part of this collaboration, the artificial agents should be designed to be aware of their limitations and be able to ask clarifying questions (Williams, 2018).

However, even collaborative systems rely on the development of autonomous methods which should be regulated through a set of ethical guidelines. The most relevant existing guidelines can be found in the ethics of algorithmic journalism. Dörr and Hollnbuchner (2017) analyze and provide an overview of commonly found ethical issues of the automatic content generation in journalism, based on a framework by Weischenberg et al. (2006) and using the ethical system introduced by Pürer (1992). While this analysis is mostly concerned with rule-based NLG, it addresses many of the issues of neural NLG. For example, Dörr and Hollnbuchner state that two of the most commonly seen issues are (1) the hallucination of information, and (2) missing information that leads to uninformed or biased outputs. Both of these problems frequently occur with neural NLG. They argue that an NLG system is a relatively autonomous actor with a delegated (moral) agency, which means that the organization that uses the system is responsible for its ethical behavior. Moreover, each journalist is responsible for their moral actions, guided by the principles of objectiveness, transparency, and accuracy. That means that they need to be able to supervise the systems that automate part of their duties to ensure that their

outcomes comply with these principles. The requirement for supervision further motivates CSI-like approaches where journalists retain the agency over the automated process.

This analysis uncovers two challenges. We need to ensure that an autonomous agent is complying with the ethical principles on a global level. As a result, we need affordances in the model that assist the individuals in ensuring that the model is exhibiting ethical behavior. The supervision of the autonomous agent by a human agent requires the careful design of mechanisms for explainability, as discussed in Chapters 3-6. To analyze and verify the compliance on the global level, Smiley et al. (2017) identify potential consequences across four categories: Human consequences, data issues, generation issues, and provenance. These categories capture whether the generated text disadvantages anyone or violates anyone's privacy by producing otherwise not disclosed information about them. It identifies the bias in the training data and problematic generations in style and content. It further evaluates whether the methods and the data are appropriately disclosed.

To capture the potential consequences of applying a model, Smiley et al. argue for the application of a checklist. However, many of its questions arise from the specific model that is being applied. As a result, this suggestion requires the creation of a customized checklist for each model. Such checklists require additional expertise, which leads to the question who is creating them. In follow-up work, Leidner and Plachouras (2017) discuss whether ethics review boards (ERB), similar to those review boards related to human-subjects studies (Enfield and Truwit, 2008), could be in charge of the checklists and ensure that a suggested research project does not lead to unintended consequences. They further point out that the same project may need to validated at multiple stages in a corporate development process: during the research phase, product development, and deployment.

The iterative nature of validation, along with the project- and model-dependent checklists, leads to a highly resource-intensive process that only some large institutions can afford.<sup>1</sup> However, the whole field of AI research could benefit from more oversight and guidance on the ethical permissibility of

---

<sup>1</sup>Some large companies already have ERBs in place.

research proposals. It is thus an open issue for the community of AI researchers to decide whether to create and require ERBs, which rules they impose, and whether they will be run by a central entity or individual institutions.

Another challenge of ethical oversight is to agree on the principles that should guide the evaluation for ethical permissibility of NLG systems. As a first attempt to summarize the concerns by [Smiley et al.](#) and those mentioned throughout this dissertation, we suggest the following three principles for NLG:

**Harmless** No text-generating system should be able to generate output that can lead to harm being done to any person or entity.

**Fair and Bias-free** A text-generating system should treat each individual, group, and community fairly and without bias pertaining to their person or occupation.

**Controlled Autonomy** A text-generating system should have moderation mechanisms that enable a person to overwrite, control, and dismiss its output if it violates one of the other two principles.

As discussed throughout this dissertation, all current models violate the first two points and could thus be seen as unethical, even if unintentionally so. We, therefore, introduce the third point to enable the intervention in harmful output and specify that the violation has to be intentional. It remains an open challenge how to address the unintentional violations of the first two principles across models and tasks. However, research should aim to minimize the violations of the first two points by developing technology that mitigates biases in neural models. Moreover, any research on something that is a dual-use technology should be accompanied by efforts to prevent its abuse.

As the potential for misuse of NLG technology grows, we require stronger prevention mechanisms that aim to detect this abuse. For example, if the output of a neural network had a detectable signature, similar to a fingerprint, we could verify that a text was not generated by that model. Much recent

research aims to solve this problem, for example, by training classifiers that distinguish between generated and human-written text (Zellers et al., 2019, Bakhtin et al., 2019).

As an alternative approach, we applied the design of collaborative systems to the problem and developed an interface that assists humans in deciding whether a text was generated (Gehrmann et al., 2019c). We demonstrated that through collaboration, we were able to educate people about the abuse of language models, addressing the education issue pointed out by Fort and Couillaud. A human-subjects study showed that it improved the human detection ability from 54%, barely above random chance, to over 72%. While GLTR does not solve the issue of neural fake news, the combination of autonomous and collaborative systems can be used to address the abuse of language models.

Similar collaborative techniques might be able to help identify harmful inductive biases in models and address some of the other ethical issues that arise from powerful NLG models.

### 9.3 COLLABORATION IN OTHER DOMAINS

While this dissertation is focused on use cases in NLG, the collaboration principles can be applied to other domains that benefit from the generation of content with neural networks. A field that uses very similar models is computational biology, specifically the computational modeling of proteins. Proteins are typically represented as strings of characters. Large language models for proteins can learn biological structures and functions (Rives et al., 2019). That means that there is potential for collaborative tools for the detection of proteins with specific properties that can be used to accelerate drug development (Sercu et al., 2019).

Another domain with applications of collaborative models is computational creativity. For example, Bau et al. (2019) developed a collaborative tool where a painter can semantically interact in an image by drawing “concepts”. For instance, if artists want to add a tree to the image, they draw the concept of a tree on the image, but the model inserts its version of a tree. This comparison relates to

the theory of forms in philosophy where a physical manifestation is not as real as its intangible idea. In their current form, neural networks are merely presented with numerous manifestations to learn the “typical” look of an object. However, they fail at learning the concept, as can be shown by their failure to paint objects in environments where they are typically not seen (Bau et al., 2019). Continuing the argument from above, this leads to major ethical concerns about these tools, as historical paintings depict a biased view of the world. Biases in datasets lead to discrepancies between the classification accuracy of people of different race or gender in other computer vision tasks (Buolamwini and Gebru, 2018). It will thus require much more work to truly be able to paint in concepts.

One can imagine many other domains where collaboration can benefit end users more than full automation and where the retention of human agency is crucial to the success of a task. However, it will only be possible to imagine medical or financial applications once these ethical concerns are appropriately addressed.

#### 9.4 CONCLUSION

Throughout this dissertation, we have argued for deep neural networks that are designed as partners to humans instead of autonomous agents. Such collaboration is characterized by the ability to explain a system’s prediction and to adjust the prediction in response to feedback from a person. However, current deep neural architectures are not designed in ways that enable them to collaborate. We introduced two necessary conditions for collaboration: interpretability and controllability.

An interpretable model is one that can explain its prediction in a way that a human can understand. It is challenging to develop explanations that mesh with the mental models that humans have of a problem. We identified in a case study of patient phenotyping that the meshing of model behavior and human understanding is especially important when the human end users do not agree with a prediction by the model. We showed that this challenge can be addressed through interaction, where

humans do not only observe a single explanation but instead get to change inputs and constrain outputs of the model. We demonstrated in two case studies of interactive systems that such interactions enable people to develop a mental model of a machine learning model’s behavior and helps them understand the limitations of the model.

We next proposed using discrete latent variables as a way to make models controllable. Models have to reason over all potential values that the variable can have, and use the value in any downstream reasoning it. By constraining or overwriting the value of the latent variable, humans can control the output of the model without changing the inputs. The latent variables can even be used to explicitly model an aspect of a problem that the standard model struggles to account for. We demonstrated that this approach can lead to better performance of a summarization model.

We further showed how to tie the latent variables to semantic interactions within an interface and developed a collaborative summarization interface. This tool represents one of the first collaborative deep learning interfaces.

Moving forward, we need to solve many more challenges to enable collaborative interface across all potential use cases. Most importantly, the methods underlying the deep latent variable models have only recently become expressive and stable enough to be used in real-world applications. Many more algorithmic and optimization improvements are required for them to become general-purpose techniques that can be applied to any model. Moreover, there are still many issues related to autonomous text generation that need to be solved, for example, the evaluation issue.

In light of the ethical challenges surrounding the training of machine learning models on corpora that comprise undesired biases, much more work is required to develop policies surrounding these powerful models that lead to ethical applications and prevent malicious use. Further, future work must introduce methods to prevent NLG models from generating unethical output that contains, for instance, factual mistakes or biased representations.

# References

- Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. [Tensorflow: Large-scale machine learning on heterogeneous distributed systems](#).
- Jaeger P Ackerman, Daniel C Bartos, Jamie D Kapplinger, David J Tester, Brian P Delisle, and Michael J Ackerman. 2016. [The promise and peril of precision medicine: phenotyping still matters most](#). In *Mayo Clinic Proceedings*, volume 91, pages 1606–1616. Elsevier.
- Julius Adebayo, Justin Gilmer, Ian Goodfellow, Moritz Hardt, and Been Kim. 2018. [Sanity checks for saliency maps](#). In *advances in neural information processing systems*.
- David Alvarez-Melis and Tommi S. Jaakkola. 2017. [A causal framework for explaining the predictions of black-box sequence-to-sequence models](#). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 412–421.
- Saleema Amershi, Maya Cakmak, William Bradley Knox, and Todd Kulesza. 2014. [Power to the people: The role of humans in interactive machine learning](#). *AI Magazine*, 35(4):105–120.
- Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Journey, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N. Bennett, Kori Inkpen, Jaime Teevan, Ruth Kikin-Gil, and Eric Horvitz. 2019. [Guidelines for human-AI interaction](#). In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, CHI ’19, pages 3:1–3:13, New York, NY, USA. ACM*.
- Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan C. Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Erich Elsen, Jesse Engel, Linxi Fan, Christopher Fougner, Tony Han, Awni Y. Hannun, Billy Jun, Patrick LeGresley, Libby Lin, Sharan Narang, Andrew Y. Ng, Sherjil Ozair, Ryan Prenger, Jonathan Raiman, Sanjeev Satheesh,

David Seetapun, Shubho Sengupta, Yi Wang, Zhiqian Wang, Chong Wang, Bo Xiao, Dani Yogatama, Jun Zhan, and Zhenyao Zhu. 2016. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*, pages 173–182.

Ashwin N Ananthakrishnan, Tianxi Cai, Guergana Savova, Su-Chun Cheng, Pei Chen, Raul Guzman Perez, Vivian S Gainer, Shawn N Murphy, Peter Szolovits, Zongqi Xia, et al. 2013. Improving case definition of crohn’s disease and ulcerative colitis in electronic medical records using natural language processing: a novel informatics approach. *Inflammatory bowel diseases*, 19(7):1411.

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6077–6086.

Valerie Anderson and Suzanne Hidi. 1988. Teaching students to summarize. *Educational leadership*, 46(4):26–28.

Kenneth C Arnold, Krysta Chauncey, and Krzysztof Z Gajos. 2018. Sentiment bias in predictive text recommendations results in biased writing. In *Proceedings of Graphics Interface*, pages 33–40.

Alan R Aronson and François-Michel Lang. 2010. An overview of metamap: historical perspective and recent advances. *Journal of the American Medical Informatics Association*, 17(3):229–236.

Leila Arras, Franziska Horn, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. 2016. Explaining predictions of non-linear classifiers in NLP. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 1–7.

Leila Arras, Franziska Horn, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. 2017. ” what is relevant in a text document? ”: An interpretable machine learning approach. *PloS one*, 12(8):e0181142.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *Advances in Neural Information Processing Systems - Deep Learning Symposium*.

Tamara Babaian, Barbara J Grosz, and Stuart M Shieber. 2002. A writer’s collaborative assistant. In *Proceedings of the 7th international conference on Intelligent user interfaces*, pages 7–14. ACM.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Anton Bakhtin, Sam Gross, Myle Ott, Yuntian Deng, Marc'Aurelio Ranzato, and Arthur Szlam. 2019. [Real or fake? learning to discriminate machine from human generated text](#). *arXiv preprint arXiv:1906.03351*.

Jonathan Bates, Samah J Fodeh, Cynthia A Brandt, and Julie A Womack. 2016. Classification of radiology reports for falls in an hiv study cohort. *Journal of the American Medical Informatics Association*, 23(e1):e113–e117.

David Bau, Jun-Yan Zhu, Hendrik Strobelt, Bolei Zhou, Joshua B. Tenenbaum, William T. Freeman, and Antonio Torralba. 2019. [GAN dissection: Visualizing and understanding generative adversarial networks](#). In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017a. [What do neural machine translation models learn about morphology?](#) In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 861–872.

Yonatan Belinkov and James Glass. 2019. [Analysis methods in neural language processing: A survey](#). *Transactions of the Association for Computational Linguistics*, 7:49–72.

Yonatan Belinkov, Lluís Màrquez, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2017b. [Evaluating layers of representation in neural machine translation on part-of-speech and semantic tagging tasks](#). In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. [A neural probabilistic language model](#). *Journal of machine learning research*, 3(Feb):1137–1155.

Michael S Bernstein, Greg Little, Robert C Miller, Björn Hartmann, Mark S Ackerman, David R Karger, David Crowell, and Katrina Panovich. 2010. [Soylent: a word processor with a crowd inside](#). In *Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology*, pages 313–322. ACM.

Olivier Bodenreider. 2004. [The unified medical language system \(UMLS\): integrating biomedical terminology](#). *Nucleic acids research*, 32(suppl 1):D267–D270.

Svetlin Bostandjiev, John O'Donovan, and Tobias Höllerer. 2012. [Tasteweights: a visual interactive hybrid recommender system](#). In *Proceedings of the sixth ACM conference on Recommender systems*, pages 35–42. ACM.

Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. 2012. [Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription](#). In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress.

Peter Brown, John Cocke, S Della Pietra, V Della Pietra, Frederick Jelinek, Robert Mercer, and Paul Roossin. 1988. *A statistical approach to language translation*. In *Proceedings of the 12th conference on Computational linguistics-Volume 1*, pages 71–76. Association for Computational Linguistics.

Miles Brundage, Shahar Avin, Jack Clark, Helen Toner, Peter Eckersley, Ben Garfinkel, Allan Dafoe, Paul Scharre, Thomas Zeitzoff, Bobby Filar, Hyrum Anderson, Heather Roff, Gregory C. Allen, Jacob Steinhardt, Carrick Flynn, Seán Ó hÉigearaigh, Simon Beard, Haydn Belfield, Sebastian Farquhar, Clare Lyle, Rebecca Crootof, Owain Evans, Michael Page, Joanna Bryson, Roman Yampolskiy, and Dario Amondei. 2018. *The malicious use of artificial intelligence: Forecasting, prevention, and mitigation*. *arXiv preprint arXiv:1802.07228*.

Duy Duc An Bui, Guilherme Del Fiol, John F Hurdle, and Siddhartha Jonnalagadda. 2016. *Extractive text summarization system to aid data extraction from full text in systematic review development*. *Journal of biomedical informatics*, 64:265–272.

Joy Buolamwini and Timnit Gebru. 2018. *Gender shades: Intersectional accuracy disparities in commercial gender classification*. In *FAT*.

Timothy J Buschman and Earl K Miller. 2007. *Top-down versus bottom-up control of attention in the prefrontal and posterior parietal cortices*. *science*, 315(5820):1860–1862.

Aylin Caliskan, Joanna J Bryson, and Arvind Narayanan. 2017. *Semantics derived automatically from language corpora contain human-like biases*. *Science*, 356(6334):183–186.

David S Carrell, Scott Halgrim, Diem-Thy Tran, Diana SM Buist, Jessica Chubak, Wendy W Chapman, and Guergana Savova. 2014. *Using natural language processing to improve efficiency of manual chart abstraction in research: the case of breast cancer recurrence*. *American journal of epidemiology*, page kwt441.

Robert J Carroll, Will K Thompson, Anne E Eyler, Arthur M Mandelin, Tianxi Cai, Raquel M Zink, Jennifer A Pacheco, Chad S Boomershine, Thomas A Lasko, Hua Xu, et al. 2012. *Portability of an algorithm to identify rheumatoid arthritis in electronic health records*. *Journal of the American Medical Informatics Association*, 19(e1):e162–e169.

Shan Carter, Zan Armstrong, Ludwig Schubert, Ian Johnson, and Chris Olah. 2019. *Activation atlas*. *Distill*. [Https://distill.pub/2019/activation-atlas](https://distill.pub/2019/activation-atlas).

Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. 2015. *Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission*. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1721–1730. ACM.

Dylan Cashman, Genevieve Patterson, Abigail Mosca, and Remco Chang. 2018. *RNNbow: Visualizing learning via backpropagation gradients in RNNs*. *IEEE Computer Graphics and Applications*, 38(6):39–50.

Leo A Celi, Peter Charlton, Mohammad M. Ghassemi, Alistair Johnson, Matthieu Komorowski, Dominic Marshall, Tristan Naumann, Kenneth Paik, Tom J. Pollard, Jesse Raffa, and Justin Salciccioli. 2016. *Secondary Analysis of Electronic Health Records*. Springer.

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1662–1675.

Junghoon Chae, Shang Gao, Arvind Ramanthan, Chad Steed, and Georgia D Tourassi. 2017. Visualization for classification in deep neural networks. In *Workshop on Visual Analytics for Deep Learning*.

Dustin Charles, Meghan Gabriel, and Michael F Furukawa. 2013. Adoption of electronic health record systems among us non-federal acute care hospitals: 2008-2012. *ONC data brief*, 9:1–9.

Lu Chen, Uta Guo, Lijo C Illipparambil, Matt D Netherton, Bhairavi Sheshadri, Eric Karu, Stephen J Peterson, and Parag H Mehta. 2016. Racing against the clock: Internal medicine residents' time spent on electronic health records. *Journal of graduate medical education*, 8(1):39–44.

Qifeng Chen and Vladlen Koltun. 2017. Photographic image synthesis with cascaded refinement networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1511–1520.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 675–686.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 551–561.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 484–494.

Travers Ching, Daniel S Himmelstein, Brett K Beaulieu-Jones, Alexandr A Kalinin, Brian T Do, Gregory P Way, Enrico Ferrero, Paul-Michael Agapow, Michael Zietz, Michael M Hoffman, et al. 2018. Opportunities and obstacles for deep learning in biology and medicine. *Journal of The Royal Society Interface*, 15(141):20170387.

Sumit Chopra, Michael Auli, and Alexander M Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98.

- Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. [A discourse-aware attention model for abstractive summarization of long documents](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 615–621.
- David A Cohn, Zoubin Ghahramani, and Michael I Jordan. 1996. [Active learning with statistical models](#). *Journal of artificial intelligence research*, 4:129–145.
- Ronan Collobert and Jason Weston. 2008. [A unified architecture for natural language processing: Deep neural networks with multitask learning](#). In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.
- Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. [Natural language processing \(almost\) from scratch](#). *Journal of Machine Learning Research*, 12(Aug):2493–2537.
- Gregory F Cooper, Constantin F Aliferis, Richard Ambrosino, John Aronis, Bruce G Buchanan, Richard Caruana, Michael J Fine, Clark Glymour, Geoffrey Gordon, Barbara H Hanusa, et al. 1997. [An evaluation of machine-learning methods for predicting pneumonia mortality](#). *Artificial intelligence in medicine*, 9(2):107–138.
- Mark W Craven and Jude W Shavlik. 1997. [Using neural networks for data mining](#). *Future generation computer systems*, 13(2-3):211–229.
- R Jordon Crouser and Remco Chang. 2012. [An affordance-based framework for human computation and human-computer collaboration](#). *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2859–2868.
- Andrew M Dai and Quoc V Le. 2015. [Semi-supervised sequence learning](#). In *Advances in Neural Information Processing Systems*, pages 3079–3087.
- Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, Anthony Bau, and James Glass. 2019. [What is one grain of sand in the desert? analyzing individual neurons in deep NLP models](#). In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.
- Misha Denil, Alban Demiraj, and Nando de Freitas. 2014. [Extraction of salient sentences from labelled documents](#). *arXiv preprint arXiv:1412.6815*.
- Joshua C Denny, Neesha N Choma, Josh F Peterson, Randolph A Miller, Lisa Bastarache, Ming Li, and Neeraja B Peterson. 2012. [Natural language processing improves identification of colorectal cancer testing in the electronic medical record](#). *Medical Decision Making*, 32(1):188–197.
- Joshua C Denny, Plomarz R Irani, Firas H Wehbe, Jeffrey D Smithers, and Anderson Spickard III. 2003. [The KnowledgeMap project: development of a concept-based medical school curriculum database](#). In *AMIA*.

Franck Dernoncourt, Ji Young Lee, Ozlem Uzuner, and Peter Szolovits. 2017. [De-identification of patient notes with recurrent neural networks](#). *Journal of the American Medical Informatics Association*, 24(3):596–606.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Debadeepa Dey, Varun Ramakrishna, Martial Hebert, and J Andrew Bagnell. 2015. [Predicting multiple structured visual interpretations](#). In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2947–2955.

Yanzhuo Ding, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. [Visualizing and understanding neural machine translation](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1150–1159.

Alexander Dlikman and Mark Last. 2016. [Using machine learning methods and linguistic features in single-document extractive summarization](#). In *Proceedings of the Workshop on Interactions between Data Mining and Natural Language Processing, DMNLP 2016, co-located with the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, ECML-PKDD 2016, Riva del Garda, Italy, September 23, 2016.*, pages 1–8.

Bonnie Dorr, David Zajic, and Richard Schwartz. 2003. [Hedge trimmer: A parse-and-trim approach to headline generation](#). In *Proceedings of the HLT-NAACL 03 on Text summarization workshop-Volume 5*, pages 1–8. Association for Computational Linguistics.

Konstantin Nicholas Dörr and Katharina Hollnbuchner. 2017. [Ethical challenges of algorithmic journalism](#). *Digital journalism*, 5(4):404–419.

Finale Doshi-Velez and Been Kim. 2017. [Towards a rigorous science of interpretable machine learning](#). *arXiv preprint arXiv:1702.08608*.

Leon E Dostert. 1955. [The georgetown-IBM experiment](#). *Machine translation of languages*. John Wiley & Sons, New York, pages 124–135.

Greg Durrett, Taylor Berg-Kirkpatrick, and Dan Klein. 2016. [Learning-based single-document summarization with compression and anaphoricity constraints](#). In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1998–2008.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2020. [Evaluating the state-of-the-art of end-to-end natural language generation: The E2E NLG challenge](#). *Computer Speech & Language*, 59:123–156.

- Henry Elder, Jennifer Foster, James Barry, and Alexander O'Connor. 2019. [Designing a symbolic intermediate representation for neural surface realization](#). In *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*, pages 65–73, Minneapolis, Minnesota. Association for Computational Linguistics.
- Jeffrey L Elman. 1990. [Finding structure in time](#). *Cognitive science*, 14(2):179–211.
- Alex Endert, Patrick Fiaux, and Chris North. 2012. [Semantic interaction for visual text analytics](#). In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 473–482. ACM.
- Alexander Endert, William Ribarsky, Cagatay Turkay, BL William Wong, Ian Nabney, I Díaz Blanco, and Fabrice Rossi. 2017. [The state of the art in integrating machine learning into visual analytics](#). In *Computer Graphics Forum*, volume 36, pages 458–486. Wiley Online Library.
- Kyle B Enfield and Jonathon D Truwit. 2008. [The purpose, composition, and function of an institutional review board: balancing priorities](#). *Respiratory care*, 53(10):1330–1336.
- Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. [Why does unsupervised pre-training help deep learning?](#) *Journal of Machine Learning Research*, 11(Feb):625–660.
- Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2009. [Visualizing higher-layer features of a deep network](#). Technical report, University of Montreal.
- Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. 2017. [Dermatologist-level classification of skin cancer with deep neural networks](#). *Nature*, 542(7639):115–118.
- Joerg Evermann, Jana-Rebecca Rehse, and Peter Fettke. 2017. [Predicting process behaviour using deep learning](#). *Decision Support Systems*, pages –.
- Matan Eyal, Tal Baumel, and Michael Elhadad. 2019. [Question answering as an automatic evaluation metric for news article summarization](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3938–3948.
- Jerry Alan Fails and Dan R Olsen Jr. 2003. [Interactive machine learning](#). In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 39–45. ACM.
- Angela Fan, Mike Lewis, and Yann Dauphin. 2018. [Hierarchical neural story generation](#). In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898.
- Angela Fan, Mike Lewis, and Yann Dauphin. 2019. [Strategies for structuring story generation](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2650–2660, Florence, Italy. Association for Computational Linguistics.

Thiago Castro Ferreira, Iacer Calixto, Sander Wubben, and Emiel Krahmer. 2017. [Linguistic realisation as machine translation: Comparing different MT models for AMR-to-text generation](#). In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 1–10.

Terrence Fong, Charles Thorpe, and Charles Baur. 2003. [Collaboration, dialogue, human-robot interaction](#). In *Robotics Research*, pages 255–266. Springer.

Karën Fort and Alain Couïnault. 2016. [Yes, we care! results of the ethics and natural language processing surveys](#). In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 1593–1600.

Carol Friedman, Philip O Alderson, John HM Austin, James J Cimino, and Stephen B Johnson. 1994. [A general natural-language text processor for clinical radiology](#). *Journal of the American Medical Informatics Association*, 1(2):161–174.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. [AllenNLP: A deep semantic natural language processing platform](#). In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6.

Albert Gatt and Emiel Krahmer. 2018. [Survey of the state of the art in natural language generation: Core tasks, applications and evaluation](#). *Journal of Artificial Intelligence Research*, 61:65–170.

S Gehrmann, H Strobelt, R Kruger, H Pfister, and AM Rush. 2019a. [Visual interaction with deep learning models through collaborative semantic inference](#). *IEEE transactions on visualization and computer graphics*.

Sebastian Gehrmann, Falcon Dai, Henry Elder, and Alexander Rush. 2018a. [End-to-end content and plan selection for data-to-text generation](#). In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 46–56.

Sebastian Gehrmann, Franck Dernoncourt, Yeran Li, Eric T Carlson, Joy T Wu, Jonathan Welt, John Foote Jr, Edward T Moseley, David W Grant, Patrick D Tyler, et al. 2018b. [Comparing deep learning and concept extraction based methods for patient phenotyping from clinical narratives](#). *PLoS one*, 13(2):e0192360.

Sebastian Gehrmann, Steven Layne, and Franck Dernoncourt. 2019b. [Improving human text comprehension through semi-markov crf-based neural section title generation](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1677–1688.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander Rush. 2019c. [GLTR: Statistical detection and visualization of generated text](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 111–116, Florence, Italy. Association for Computational Linguistics.

Felix A Gers and E Schmidhuber. 2001. **LSTM recurrent networks learn simple context-free and context-sensitive languages.** *IEEE Transactions on Neural Networks*, 12(6):1333–1340.

James J Gibson. 1977. **The theory of affordances.** *Hilldale, USA*, 1:2.

Yoav Goldberg. 2016. **A primer on neural network models for natural language processing.** *Journal of Artificial Intelligence Research*, 57:345–420.

Bryce Goodman and Seth Flaxman. 2017. **European union regulations on algorithmic decision-making and a “right to explanation”.** *AI Magazine*, 38(3):50–57.

Spence Green, Sida I Wang, Jason Chuang, Jeffrey Heer, Sebastian Schuster, and Christopher D Manning. 2014. **Human effort and machine learnability in computer aided translation.** In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1225–1236.

Barbara J Grosz. 1996. **Collaborative systems (AAAI-94 presidential address).** *AI magazine*, 17(2):67.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. **Incorporating copying mechanism in sequence-to-sequence learning.** In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640.

Varun Gulshan, Lily Peng, Marc Coram, Martin C Stumpe, Derek Wu, Arunachalam Narayanaswamy, Subhashini Venugopalan, Kasumi Widner, Tom Madams, Jorge Cuadros, et al. 2016. **Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs.** *JAMA*, 316(22):2402–2410.

Matthew Guzdial and Mark Riedl. 2019. **An interaction framework for studying co-creative AI.** *arXiv preprint arXiv:1903.09709*.

Matthew James Guzdial, Jonathan Chen, Shao-Yu Chen, and Mark Riedl. 2017. **A general level design editor for co-creative level design.** In *Thirteenth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Richard HR Hahnloser, Rahul Sarpeshkar, Misha A Mahowald, Rodney J Douglas, and H Sebastian Seung. 2000. **Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit.** *Nature*, 405(6789):947.

Yoni Halpern, Steven Horng, Youngduck Choi, and David Sontag. 2016. **Electronic medical record phenotyping using the anchor and learn framework.** *Journal of the American Medical Informatics Association*, page ocw011.

Hany Hassan Awadalla, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, Will Lewis, Mu Li, Shujie Liu, Tie-Yan

- Liu, Renqian Luo, Arul Menezes, Tao Qin, Frank Seide, Xu Tan, Fei Tian, Lijun Wu, Shuangzhi Wu, Yingce Xia, Dongdong Zhang, Zhirui Zhang, and Ming Zhou. 2018. *Achieving human parity on automatic chinese to english news translation*. *arXiv preprint arXiv:1803.05567*.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.
- Jeffrey Heer. 2019. Agency plus automation: Designing artificial intelligence into interactive systems. *Proceedings of the National Academy of Sciences*, 116(6):1844–1850.
- Ryan Henderson and Rasmus Rothe. 2017. Picasso: A modular framework for visualizing the learning process of neural network image classifiers. *Journal of Open Research Software*, 5(1).
- Karl Moritz Hermann, Tomas Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.
- Alex Hern. 2017. Facebook translates ‘good morning’ into ‘attack them’, leading to arrest. *The Guardian*.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Fred Hohman, Nathan Hodas, and Duen Horng Chau. 2017. Shapeshop: Towards understanding deep learning representations via interactive experimentation. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, pages 1694–1699. ACM.
- Fred Matthew Hohman, Minsuk Kahng, Robert Pienta, and Duen Horng Chau. 2018. Visual analytics in deep learning: An interrogative survey for the next frontiers. *IEEE Transactions on Visualization and Computer Graphics*.
- Kenneth Holstein, Jennifer Wortman Vaughan, Hal Daumé III, Miro Dudík, and Hanna Wallach. 2019. Improving fairness in machine learning systems: What do industry practitioners need? In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, page 600. ACM.
- Andreas Holzinger. 2016. Interactive machine learning for health informatics: when do we need the human-in-the-loop? *Brain Informatics*, 3(2):119–131.
- Eric Horvitz. 1999. Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 159–166. ACM.
- Eduard Hovy. 1987. Generating natural language under pragmatic constraints. *Journal of Pragmatics*, 11(6):689–719.

George Hripcsak and David J Albers. 2013. Next-generation phenotyping of electronic health records. *Journal of the American Medical Informatics Association*, 20(1):117–121.

George Hripcsak, John HM Austin, Philip O Alderson, and Carol Friedman. 2002. Use of natural language processing to translate clinical information from a database of 889,921 chest radiographic reports 1. *Radiology*, 224(1):157–163.

Wan-Ting Hsu, Chieh-Kai Lin, Ming-Ying Lee, Kerui Min, Jing Tang, and Min Sun. 2018. A unified model for extractive and abstractive summarization using inconsistency loss. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 132–141.

Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Ian Simon, Curtis Hawthorne, Noam Shazeer, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck. 2019. Music transformer. In *International Conference on Learning Representations*.

Michael C Hughes, Gabriel Hope, Leah Weiner, Thomas H McCoy Jr, Roy H Perlis, Erik B Suderth, and Finale Doshi-Velez. 2018. Semi-supervised prediction-constrained topic models. In *AISTATS*, pages 1067–1076.

Frederick Jelinek. 1997. *Statistical methods for speech recognition*. MIT press.

Peter B Jensen, Lars J Jensen, and Søren Brunak. 2012. Mining electronic health records: towards better research applications and clinical care. *Nature Reviews Genetics*, 13(6):395–405.

Liu Jiang, Shixia Liu, and Changjian Chen. 2019. Recent research advances on interactive machine learning. *Journal of Visualization*, 22(2):401–417.

Hongyan Jing and Kathleen R McKeown. 1999. The decomposition of human-written summary sentences. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 129–136.

Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. MIMIC-III, a freely accessible critical care database. *Scientific data*, 3.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2017. Google’s multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Philip Nicholas Johnson-Laird. 1983. *Mental models: Towards a cognitive science of language, inference, and consciousness*. 6. Harvard University Press.

- Ákos Kádár, Grzegorz Chrupała, and Afra Alishahi. 2015. [Lingusitic analysis of multi-modal recurrent neural networks](#). In *Proceedings of the Fourth Workshop on Vision and Language*, pages 8–9, Lisbon, Portugal. Association for Computational Linguistics.
- Akos Kádár, Grzegorz Chrupała, and Afra Alishahi. 2017. [Representation of linguistic form and function in recurrent neural networks](#). *Computational Linguistics*, 43(4):761–780.
- Minsuk Kahng, Pierre Y Andrews, Aditya Kalro, and Duen Horng Polo Chau. 2018. [ActiVis: Visual exploration of industry-scale deep neural network models](#). *IEEE transactions on visualization and computer graphics*, 24(1):88–97.
- Nal Kalchbrenner and Phil Blunsom. 2013. [Recurrent continuous translation models](#). In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709.
- Nang Kang, Bharat Singh, Mulligen Erik M van Afzal, Zubair and, and Jan A Kors. 2013. [Using rule-based natural language processing to improve disease normalization in biomedical text](#). *Journal of the American Medical Informatics Association*, 20:876–881.
- Ashish Kapoor, Bongshin Lee, Desney Tan, and Eric Horvitz. 2010. [Interactive optimization for steering machine classification](#). In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1343–1352. ACM.
- Andrej Karpathy, Justin Johnson, and Fei-Fei Li. 2015. [Visualizing and understanding recurrent networks](#). *arXiv preprint arXiv:1506.02078*.
- Daniel A Keim, Florian Mansmann, Jörn Schneidewind, Jim Thomas, and Hartmut Ziegler. 2008. [Visual analytics: Scope and challenges](#). In *Visual data mining*, pages 76–90. Springer.
- Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. [Ctrl: A conditional transformer language model for controllable generation](#). *arXiv preprint arXiv:1909.05858*.
- Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. [Globally coherent text generation with neural checklist models](#). In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 329–339.
- Been Kim, Rajiv Khanna, and Oluwasanmi O Koyejo. 2016a. [Examples are not enough, learn to criticize! criticism for interpretability](#). In *Advances in Neural Information Processing Systems*, pages 2280–2288.
- Yoon Kim. 2014. [Convolutional neural networks for sentence classification](#). In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.
- Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016b. [Character-aware neural language models](#). In *AAAI*, pages 2741–2749.

Yoon Kim, Sam Wiseman, and Alexander M Rush. 2018. *A tutorial on deep latent variable models of natural language*. *arXiv preprint arXiv:1812.06834*.

Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. 2019. *The (un)reliability of saliency methods*. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 267–280. Springer.

Jacqueline C Kirby, Peter Speltz, Luke V Rasmussen, Melissa Basford, Omri Gottesman, Peggy L Peissig, Jennifer A Pacheco, Gerard Tromp, Jyotishman Pathak, David S Carrell, et al. 2016. *PheKB: a catalog and workflow for creating electronic phenotype algorithms for transportability*. *Journal of the American Medical Informatics Association*, 23(6):1046–1052.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. *OpenNMT: Open-source toolkit for neural machine translation*. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72.

Robert P Kocher and Eli Y Adashi. 2011. *Hospital readmissions and the affordable care act: paying for coordinated quality care*. *Jama*, 306(16):1794–1795.

Philipp Koehn and Rebecca Knowles. 2017. *Six challenges for neural machine translation*. *Association for Computational Linguistics*, page 28.

Shuoyang Ding Hainan Xu Philipp Koehn. 2019. *Saliency-driven word alignment interpretation for neural machine translation*. *WMT 2019*, page 1.

Pang Wei Koh and Percy Liang. 2017. *Understanding black-box predictions via influence functions*. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1885–1894. JMLR.org.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. *Neural AMR: Sequence-to-sequence models for parsing and generation*. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 146–157.

Josua Krause, Aritra Dasgupta, Jordan Swartz, Yindalon Aphinyanaphongs, and Enrico Bertini. 2017. *A workflow for visual diagnostics of binary classifiers using instance-level explanations*. In *2017 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 162–172. IEEE.

Josua Krause, Adam Perer, and Kenney Ng. 2016. *Interacting with predictions: Visual inspection of black-box machine learning models*. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 5686–5697. ACM.

Joseph B Kruskal. 1964. *Nonmetric multidimensional scaling: a numerical method*. *Psychometrika*, 29(2):115–129.

Todd Kulesza, Margaret Burnett, Weng-Keen Wong, and Simone Stumpf. 2015. [Principles of explanatory debugging to personalize interactive machine learning](#). In *Proceedings of the 20th international conference on intelligent user interfaces*, pages 126–137. ACM.

Todd Kulesza, Simone Stumpf, Margaret Burnett, Weng-Keen Wong, Yann Riche, Travis Moore, Ian Oberst, Amber Shinsel, and Kevin McIntosh. 2010. [Explanatory debugging: Supporting end-user debugging of machine-learned programs](#). In *Visual Languages and Human-Centric Computing (VL/HCC), 2010 IEEE Symposium on*, pages 41–48. IEEE.

Bum Chul Kwon, Min-Je Choi, Joanne Taery Kim, Edward Choi, Young Bin Kim, Soonwook Kwon, Jimeng Sun, and Jaegul Choo. 2019. [RetainVis: Visual analytics with interpretable and interactive recurrent neural networks on electronic medical records](#). *IEEE transactions on visualization and computer graphics*, 25(1):299–309.

Carmen Lacave and Francisco J Díez. 2002. [A review of explanation methods for bayesian networks](#). *The Knowledge Engineering Review*, 17(2):107–127.

Vivian Lai and Chenhao Tan. 2019. [On human predictions with explanations and predictions of machine learning models: A case study on deception detection](#). In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 29–38. ACM.

Quoc V Le, Marc’Aurelio Ranzato, Rajat Monga, Matthieu Devin, Greg Corrado, Kai Chen 0010, Jeffrey Dean, and Andrew Y Ng. 2012. [Building high-level features using large scale unsupervised learning](#). *ICML*.

Y Lecun, L Bottou, Y Bengio, and P Haffner. 1998. [Gradient-based learning applied to document recognition](#). *Proceedings of the IEEE*, 86(11):2278–2324.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. [Gradient-based learning applied to document recognition](#). *Proceedings of the IEEE*, 86(11):2278–2324.

Jaesong Lee, Joong-Hwi Shin, and Jun-Seok Kim. 2017. [Interactive visualization and manipulation of attention-based neural machine translation](#). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 121–126.

Tao Lei, Regina Barzilay, and Tommi S. Jaakkola. 2016. [Rationalizing neural predictions](#). In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 107–117.

Jochen L Leidner and Vassilis Plachouras. 2017. [Ethical by design: ethics best practices for natural language processing](#). In *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*, pages 30–40.

Chenliang Li, Weiran Xu, Si Li, and Sheng Gao. 2018a. [Guiding generation for abstractive text summarization based on key information guide network](#). In *Proceedings of the 2018 Conference of*

*the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 55–60.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016a. **Visualizing and Understanding Neural Models in NLP**. In *NAAACL*, pages 1–10, San Diego, California. Association for Computational Linguistics.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016b. **Understanding neural networks through representation erasure**. *arXiv preprint arXiv:1612.08220*.

Piji Li, Lidong Bing, and Wai Lam. 2018b. **Actor-critic based training framework for abstractive summarization**. *arXiv preprint arXiv:1803.11070*.

Katherine P Liao, Tianxi Cai, Vivian Gainer, Sergey Goryachev, Qing Zeng-treitler, Soumya Raychaudhuri, Peter Szolovits, Susanne Churchill, Shawn Murphy, Isaac Kohane, et al. 2010. **Electronic medical records for discovery research in rheumatoid arthritis**. *Arthritis care & research*, 62(8):1120–1127.

Katherine P Liao, Tianxi Cai, Guergana K Savova, Shawn N Murphy, Elizabeth W Karlson, Ashwin N Ananthakrishnan, Vivian S Gainer, Stanley Y Shaw, Zongqi Xia, Peter Szolovits, et al. 2015. **Development of phenotype algorithms using electronic medical records and incorporating natural language processing**. *bmj*, 350:h1885.

Brian Y Lim, Anind K Dey, and Daniel Avrahami. 2009. **Why and why not explanations improve the intelligibility of context-aware intelligent systems**. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2119–2128. ACM.

Chin-Yew Lin. 2004. **Rouge: A package for automatic evaluation of summaries**. In *Text summarization branches out*, pages 74–81.

Halden Lin, Tongshuang Wu, Kanit Wongsuphasawat, Yejin Choi, and Jeffrey Heer. 2018. **Visualizing attention in sequence-to-sequence summarization models**. *Visual Analytics Science and Technologies Poster Program*.

Todd Lingren, Pei Chen, Joseph Bochenek, Finale Doshi-Velez, Patty Manning-Courtney, Julie Bickel, Leah Wildenger Welchons, Judy Reinhold, Nicole Bing, Yizhao Ni, et al. 2016. **Electronic health record based algorithm to identify patients with autism spectrum disorder**. *PLoS one*, 11(7):e0159621.

Zachary C. Lipton. 2018. **The mythos of model interpretability**. *Communications of the ACM*, 61(10):36–43.

Fei Liu, Nicole Lee Fella, and Kexin Liao. 2016. **Modeling language vagueness in privacy policies using deep neural networks**. In *2016 AAAI Fall Symposium Series*.

Peter J. Liu\*, Mohammad Saleh\*, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. In *International Conference on Learning Representations*.

Shixia Liu, Xiting Wang, Mengchen Liu, and Jun Zhu. 2017. Towards better analysis of machine learning models: A visual analytics perspective. *Visual Informatics*, 1(1):48–56.

Shusen Liu, Zhimin Li, Tao Li, Vivek Srikumar, Valerio Pascucci, and Peer-Timo Bremer. 2019. NL-IZE: A perturbation-driven visual interrogation tool for analyzing and interpreting natural language inference models. *IEEE transactions on visualization and computer graphics*, 25(1):651–660.

Yafeng Lu, Rolando Garcia, Brett Hansen, Michael Gleicher, and Ross Maciejewski. 2017. The state-of-the-art in predictive visual analytics. In *Computer Graphics Forum*, volume 36, pages 539–562. Wiley Online Library.

Yi Luan, Shinji Watanabe, and Bret Harsham. 2015. Efficient learning for spoken language understanding tasks with word embedding based pre-training. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research*, 9(Nov):2579–2605.

Saad Mahamood and Ehud Reiter. 2011. Generating affective natural language for parents of neonatal infants. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 12–21.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Cettolo Mauro, Girardi Christian, and Federico Marcello. 2012. Wit3: Web inventory of transcribed and translated talks. In *Conference of European Association for Machine Translation*, pages 261–268.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6294–6305.

Tomas Mikolov, Martin Karafiat, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Interspeech*, volume 2, page 3.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

- Yao Ming, Shaozu Cao, Ruixiang Zhang, Zhen Li, Yuanzhe Chen, Yangqiu Song, and Huamin Qu. 2017. [Understanding hidden memories of recurrent neural networks](#). In *2017 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 13–24. IEEE.
- Marvin Minsky and Seymour A Papert. 1969. *Perceptrons: An introduction to computational geometry*. MIT press.
- Travis B Murdoch and Allan S Detsky. 2013. [The inevitable application of big data to health care](#). *Jama*, 309(13):1351–1352.
- Prakash M Nadkarni, Lucila Ohno-Machado, and Wendy W Chapman. 2011. [Natural language processing: an introduction](#). *Journal of the American Medical Informatics Association*, 18(5):544–551.
- Vinod Nair and Geoffrey E Hinton. 2010. [Rectified linear units improve restricted boltzmann machines](#). In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.
- Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. [Summarunner: A recurrent neural network based sequence model for extractive summarization of documents](#). In *AAAI*, pages 3075–3081.
- Ramesh Nallapati, Bowen Zhou, and Mingbo Ma. 2016a. [Classify or select: Neural architectures for extractive document summarization](#). *arXiv preprint arXiv:1611.04244*.
- Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016b. [Abstractive text summarization using sequence-to-sequence rnns and beyond](#). In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290.
- Menaka Narayanan, Emily Chen, Jeffrey He, Been Kim, Sam Gershman, and Finale Doshi-Velez. 2018. [How do humans understand explanations from machine learning systems? an evaluation of the human-interpretability of explanation](#). *arXiv preprint arXiv:1802.00682*.
- Anh Nguyen, Jason Yosinski, and Jeff Clune. 2015. [Deep neural networks are easily fooled: High confidence predictions for unrecognizable images](#). In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 427–436.
- Augustus Odena, Vincent Dumoulin, and Chris Olah. 2016. [Deconvolution and checkerboard artifacts](#). *Distill*.
- Chris Olah and Shan Carter. 2016. [Attention and augmented recurrent neural networks](#). *Distill*.
- Chris Olah, Arvind Satyanarayan, Ian Johnson, Shan Carter, Ludwig Schubert, Katherine Ye, and Alexander Mordvintsev. 2018. [The building blocks of interpretability](#). *Distill*, 3(3):e10.
- Paul Over, Hoa Dang, and Donna Harman. 2007. [Duc in context](#). *Information Processing & Management*, 43(6):1506–1520.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. [Bleu: a method for automatic evaluation of machine translation](#). In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Deokgun Park, Seungyeon Kim, Jurim Lee, Jaegul Choo, Nicholas Diakopoulos, and Niklas Elmquist. 2018. [Conceptvector: text visual analytics via interactive lexicon building using word embedding](#). *IEEE transactions on visualization and computer graphics*, 24(1):361–370.

Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. 2019. [Semantic image synthesis with spatially-adaptive normalization](#). In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2337–2346.

Ramakanth Pasunuru and Mohit Bansal. 2018. [Multi-reward reinforced summarization with saliency and entailment](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 646–653.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. [A deep reinforced model for abstractive summarization](#). In *International Conference on Learning Representations*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. [Glove: Global vectors for word representation](#). In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

RH Perlis, DV Iosifescu, VM Castro, SN Murphy, VS Gainer, Jessica Minnier, T Cai, S Goryachev, Q Zeng, PJ Gallagher, et al. 2012. [Using electronic medical records to enable large-scale studies in psychiatry: treatment resistant depression as a model](#). *Psychological medicine*, 42(01):41–50.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. [Semi-supervised sequence tagging with bidirectional language models](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1756–1765.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Nicola Pezzotti, Thomas Höllt, Jan Van Gemert, Boudeijn PF Lelieveldt, Elmar Eisemann, and Anna Vilanova. 2018. [Deepeyes: Progressive visual analytics for designing deep neural networks](#). *IEEE transactions on visualization and computer graphics*, 24(1):98–108.

Rimma Pivarov and Noémie Elhadad. 2015. [Automated methods for the summarization of electronic health records](#). *Journal of the American Medical Informatics Association*, 22(5):938–947.

Sameer Pradhan, Noémie Elhadad, Brett R South, David Martinez, Lee Christensen, Amy Vogel, Hanna Suominen, Wendy W Chapman, and Guergana Savova. 2015. [Evaluating the state of the art](#)

in disorder recognition and normalization of the clinical narrative. *Journal of the American Medical Informatics Association*, 22(1):143–154.

Heinz Pürer. 1992. Ethik in journalismus und massenkommunikation. versuch einer theorie-synopse. *Publizistik*, 37(3):304–321.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).

Rajesh Ranganath, Adler Perotte, Noémie Elhadad, and David Blei. 2016. Deep survival analysis. In *Machine Learning for Healthcare Conference*, pages 101–114.

Paulo E Rauber, Samuel G Fadel, Alexandre X Falcao, and Alexandru C Telea. 2016. Visualizing the hidden activity of artificial neural networks. *IEEE transactions on visualization and computer graphics*, 23(1):101–110.

Narges Razavian, Saul Blecker, Ann Marie Schmidt, Aaron Smith-McLallen, Somesh Nigam, and David Sontag. 2015. Population-level prediction of type 2 diabetes from claims data and analysis of risk factors. *Big Data*, 3(4):277–287.

Ehud Reiter. 1994. Has a consensus NL generation architecture appeared, and is it psycholinguistically plausible? In *Proceedings of the Seventh International Workshop on Natural Language Generation*, pages 163–170. Association for Computational Linguistics.

Ehud Reiter and Robert Dale. 1997. Building applied natural language generation systems. *Natural Language Engineering*, 3(1):57–87.

Ehud Reiter and Robert Dale. 2000. *Building natural language generation systems*. Cambridge university press.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why should I trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144. ACM.

Alexander Rives, Siddharth Goyal, Joshua Meier, Demi Guo, Myle Ott, C Lawrence Zitnick, Jerry Ma, and Rob Fergus. 2019. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *bioRxiv*, page 622803.

Frank Rosenblatt. 1957. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory.

Andrew Slavin Ross, Michael C Hughes, and Finale Doshi-Velez. 2017. Right for the right reasons: training differentiable models by constraining their explanations. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 2662–2670. AAAI Press.

- Andreas Rücklé and Iryna Gurevych. 2017. End-to-end non-factoid question answering with an interactive visualization of neural attention weights. *Proceedings of ACL 2017, System Demonstrations*, pages 19–24.
- Sebastian Ruder. 2016. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.
- Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389.
- Stuart J Russell and Peter Norvig. 2016. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited.
- Dominik Sacha, Matthias Kraus, Daniel A Keim, and Min Chen. 2019. VIS4ML: An ontology for visual analytics assisted machine learning. *IEEE transactions on visualization and computer graphics*, 25(1):385–395.
- Mohammed Saeed, Christine Lieu, Greg Raber, and Roger G Mark. 2002. Mimic ii: a massive temporal icu patient database to support research in intelligent patient monitoring. In *Computers in Cardiology, 2002*, pages 641–644. IEEE.
- Evan Sandhaus. 2008. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752.
- Raymond Francis Sarmiento and Franck Dernoncourt. 2016. Improving patient cohort identification using natural language processing. *Secondary Analysis of Electronic Health Records*, pages 405–415.
- Guergana K Savova, James J Masanz, Philip V Ogren, Jiaping Zheng, Sunghwan Sohn, Karin C Kipper-Schuler, and Christopher G Chute. 2010. Mayo clinical text analysis and knowledge extraction system (cTAKES): architecture, component evaluation and applications. *Journal of the American Medical Informatics Association*, 17(5):507–513.
- Guergana K Savova, Janet E Olson, Sean P Murphy, Victoria L Cafourek, Fergus J Couch, Matthew P Goetz, James N Ingle, Vera J Suman, Christopher G Chute, and Richard M Weinshilboum. 2012. Automated discovery of drug treatment patterns for endocrine therapy of breast cancer within an electronic medical record. *Journal of the American Medical Informatics Association*, 19(e1):e83–e89.
- Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1073–1083.

Barbara Seidlhofer. 1995. *Approaches to summarization: Discourse analysis and language education*, volume 11. Gunter Narr Verlag.

Tom Sercu, Sebastian Gehrmann, Hendrik Strobelt, Payel Das, Inkit Padhi, Cicero Dos Santos, Kahini Wadhawan, and Vijil Chenthamarakshan. 2019. Interactive visual exploration of latent space (ivels) for peptide auto-encoder model selection. *Proceedings of the ICLR 2019 Workshop on Deep Generative Models for Highly Structured Data*.

Claude Elwood Shannon. 1948. *A mathematical theory of communication*. *Bell system technical journal*, 27(3):379–423.

Ben Shneiderman and Catherine Plaisant. 2006. *Strategies for evaluating information visualization tools: multi-dimensional in-depth long-term case studies*. In *Proceedings of the 2006 AVI Workshop on BEyond time and errors: novel evaluation methods for information visualization, BELIV 2006, Venice, Italy, May 23, 2006*, pages 1–7.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. *Deep inside convolutional networks: Visualising image classification models and saliency maps*. *arXiv preprint arXiv:1312.6034*.

Charese Smiley, Frank Schilder, Vassilis Plachouras, and Jochen L Leidner. 2017. *Say the right thing right: Ethics issues in natural language generation systems*. In *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*, pages 103–108.

Daniel Smilkov, Shan Carter, D Sculley, Fernanda B Viégas, and Martin Wattenberg. 2017a. *Direct-manipulation visualization of deep networks*. *arXiv preprint arXiv:1708.03788*.

Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. 2017b. *Smoothgrad: removing noise by adding noise*. *arXiv preprint arXiv:1706.03825*.

Daniel Smilkov, Nikhil Thorat, Charles Nicholson, Emily Reif, Fernanda B Viégas, and Martin Wattenberg. 2016. *Embedding projector: Interactive visualization and interpretation of embeddings*. *arXiv preprint arXiv:1611.05469*.

Kent A Spackman, Keith E Campbell, and Roger A Côté. 1997. *SNOMED RT: a reference terminology for health care*. In *Proceedings of the AMIA annual fall symposium*, page 640. American Medical Informatics Association.

Charles D Stolper, Adam Perer, and David Gotz. 2014. *Progressive visual analytics: User-driven visual exploration of in-progress analytics*. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1653–1662.

Charles J Stone. 1984. *Classification and regression trees*. *Wadsworth International Group*, 8:452–456.

- Justin A Strauss, Chun R Chao, Marilyn L Kwan, Syed A Ahmed, Joanne E Schottinger, and Virginia P Quinn. 2013. [Identifying primary and recurrent cancers using a sas-based natural language processing algorithm](#). *Journal of the American Medical Informatics Association*, 20(2):349–355.
- Hendrik Strobelt, Sebastian Gehrmann, Michael Behrisch, Adam Perer, Hanspeter Pfister, and Alexander Rush. 2018a. [Debugging sequence-to-sequence models with Seq2Seq-Vis](#). In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 368–370.
- Hendrik Strobelt, Sebastian Gehrmann, Michael Behrisch, Adam Perer, Hanspeter Pfister, and Alexander M Rush. 2019. [Seq2Seq-Vis: A visual debugging tool for sequence-to-sequence models](#). *IEEE transactions on visualization and computer graphics*, 25(1):353–363.
- Hendrik Strobelt, Sebastian Gehrmann, Hanspeter Pfister, and Alexander M Rush. 2018b. [LSTMVis: A tool for visual analysis of hidden state dynamics in recurrent neural networks](#). *IEEE transactions on visualization and computer graphics*, 24(1):667–676.
- Amber Stubbs and Özlem Uzuner. 2015. [Annotating risk factors for heart disease in clinical narratives for diabetic patients](#). *Journal of biomedical informatics*, 58:S78–S91.
- Simone Stumpf, Vidya Rajaram, Lida Li, Weng-Keen Wong, Margaret Burnett, Thomas Dietterich, Erin Sullivan, and Jonathan Herlocker. 2009. [Interacting meaningfully with machine learning systems: Three experiments](#). *International Journal of Human-Computer Studies*, 67(8):639–662.
- Simeng Sun, Ori Shapira, Ido Dagan, and Ani Nenkova. 2019. [How to compare summarizers without target length? pitfalls, solutions and re-examination of the neural summarization literature](#). In *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*, pages 21–29.
- Weiyi Sun, Anna Rumshisky, and Ozlem Uzuner. 2013. [Annotating temporal information in clinical narratives](#). *Journal of biomedical informatics*, 46:S5–S12.
- Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. [Sequence to sequence learning with neural networks](#). In *Advances in Neural Information Processing Systems*, pages 3104–3112.
- Mirac Suzgun, Sebastian Gehrmann, Yonatan Belinkov, and Stuart M Shieber. 2019. [LSTM networks can perform dynamic counting](#). *arXiv preprint arXiv:1906.03648*.
- Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. [Abstractive document summarization with a graph-based attentional neural model](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1171–1181.
- Loren G Terveen. 1995. [Overview of human-computer collaboration](#). *Knowledge-Based Systems*, 8(2-3):67–81.

Andree Thielges, Florian Schmidt, and Simon Hegelich. 2016. [The devil's triangle: Ethical considerations on developing bot detection methods](#). In *2016 AAAI Spring Symposium Series*.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. [Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition](#). In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics.

Maxim Topaz, Kenneth Lai, Dawn Dowding, Victor J Lei, Anna Zisberg, Kathryn H Bowles, and Li Zhou. 2016. [Automated identification of wound information in clinical notes of patients with heart diseases: Developing and validating a natural language processing application](#). *International Journal of Nursing Studies*, 64:25–31.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. [Modeling coverage for neural machine translation](#). In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 76–85.

Alan Turing. 1950. [Computing machinery and intelligence](#). *Mind*, 59(236):433.

F-Y Tzeng and K-L Ma. 2005. [Opening the black box-data driven visualization of neural networks](#). In *VIS 05. IEEE Visualization, 2005.*, pages 383–390. IEEE.

Özlem Uzuner. 2009. [Recognizing obesity and comorbidities in sparse data](#). *Journal of the American Medical Informatics Association*, 16(4):561–570.

Özlem Uzuner, Ira Goldstein, Yuan Luo, and Isaac Kohane. 2008. [Identifying patient smoking status from medical discharge records](#). *Journal of the American Medical Informatics Association*, 15(1):14–24.

Özlem Uzuner, Imre Solti, and Eithon Cadag. 2010. [Extracting medication information from clinical text](#). *Journal of the American Medical Informatics Association*, 17(5):514–518.

Özlem Uzuner, Brett R South, Shuying Shen, and Scott L DuVall. 2011. [2010 i2b2/VA challenge on concepts, assertions, and relations in clinical text](#). *Journal of the American Medical Informatics Association*, 18(5):552–556.

Teun A Van Dijk. 1977. [Semantic macro-structures and knowledge frames in discourse comprehension](#). *Cognitive processes in comprehension*, 332.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#). In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Alfredo Vellido, José David Martín-Guerrero, and Paulo JG Lisboa. 2012. [Making machine learning models interpretable](#). In *ESANN*, volume 12, pages 163–172. Citeseer.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Michael Völske, Martin Potthast, Shahbaz Syed, and Benno Stein. 2017. TL; DR: Mining reddit to learn automatic summarization. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 59–63.

Sandra Wachter, Brent Mittelstadt, and Chris Russell. 2017. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harv.JL & Tech.*, 31:841.

Junpeng Wang, Liang Gou, Hao Yang, and Han-Wei Shen. 2018a. GANViz: A visual analytics approach to understand the adversarial game. *IEEE transactions on visualization and computer graphics*, 24(6):1905–1917.

Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. 2018b. High-resolution image synthesis and semantic manipulation with conditional GANs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8798–8807.

Siegfried Weischenberg, Maja Malik, and Armin Scholl. 2006. Zentrale befunde der aktuellen repräsentativbefragung deutscher journalisten. journalismus in deutschland 2005. *Media Perspektiven*, o. Jg, 7:346–361.

James Wexler, Mahima Pushkarna, Tolga Bolukbasi, Martin Wattenberg, Fernanda Viégas, and Jimbo Wilson. 2019. The what-if tool: Interactive probing of machine learning models. *IEEE transactions on visualization and computer graphics*.

Paul Rodriguez Janet Wiles. 1998. Recurrent neural networks can learn to implement symbol-sensitive counting. In *Advances in Neural Information Processing Systems 10: Proceedings of the 1997 Conference*, volume 10, page 87. MIT Press.

Yorick Wilks. 2010. Close engagements with artificial companion.

Tom Williams. 2018. Toward ethical natural language generation for human-robot interaction. In *Companion of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, pages 281–282. ACM.

Kanit Wongsuphasawat, Daniel Smilkov, James Wexler, Jimbo Wilson, Dandelion Mané, Doug Fritz, Dilip Krishnan, Fernanda B Viégas, and Martin Wattenberg. 2018. Visualizing dataflow graphs of deep learning models in tensorflow. *IEEE transactions on visualization and computer graphics*, 24(1):1–12.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto

Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. **Google’s neural machine translation system: Bridging the gap between human and machine translation.** *arXiv preprint arXiv:1609.08144*.

Yonghui Wu, Jun Xu, Min Jiang, Yaoyun Zhang, and Hua Xu. 2015. **A study of neural word embeddings for named entity recognition in clinical text.** In *AMIA Annual Symposium Proceedings*, volume 2015, page 1326. American Medical Informatics Association.

Zongqi Xia, Elizabeth Secor, Lori B Chibnik, Riley M Bove, Suchun Cheng, Tanuja Chitnis, Andrew Cagan, Vivian S Gainer, Pei J Chen, Katherine P Liao, et al. 2013. **Modeling disease severity in multiple sclerosis using electronic health records.** *PloS one*, 8(11):e78927.

Hua Xu, Shane P Stenner, Son Doan, Kevin B Johnson, Lemuel R Waitman, and Joshua C Denny. 2010. **MedEx: a medication information extraction system for clinical narratives.** *Journal of the American Medical Informatics Association*, 17(1):19–24.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. **Show, attend and tell: Neural image caption generation with visual attention.** In *International conference on machine learning*, pages 2048–2057.

Qian Yang. 2018. **Machine learning as a UX design material: How can we imagine beyond automation, recommenders, and reminders?** In *2018 AAAI Spring Symposium Series*.

Yiming Yang and Jan O Pedersen. 1997. **A comparative study on feature selection in text categorization.** In *Icml*, volume 97, pages 412–420.

Ye Ye, Michael M Wagner, Gregory F Cooper, Jeffrey P Ferraro, Howard Su, Per H Gesteland, Peter J Haug, Nicholas E Millett, John M Aronis, Andrew J Nowalk, et al. 2017. **A study of the transferability of influenza case detection systems between two large healthcare systems.** *PloS one*, 12(4):e0174970.

Ming Yin, Jennifer Wortman Vaughan, and Hanna Wallach. 2019. **Understanding the effect of accuracy on trust in machine learning models.** In *CHI ’19 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM Press.

Jason Yosinski, Jeff Clune, Thomas Fuchs, and Hod Lipson. 2015. **Understanding neural networks through deep visualization.** In *In ICML Workshop on Deep Learning*. Citeseer.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. **Recurrent neural network regularization.** *arXiv preprint arXiv:1409.2329*.

Matthew D Zeiler and Rob Fergus. 2014. **Visualizing and understanding convolutional networks.** In *European conference on computer vision*, pages 818–833. Springer.

- Matthew D Zeiler, Dilip Krishnan, Graham W Taylor, and Rob Fergus. 2010. [Deconvolutional networks](#). In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2528–2535. IEEE.
- Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. [Defending against neural fake news](#). *arXiv preprint arXiv:1905.12616*.
- Wenyuan Zeng, Wenjie Luo, Sanja Fidler, and Raquel Urtasun. 2016. [Efficient summarization with read-again and copy mechanism](#). *arXiv preprint arXiv:1611.03382*.
- Jiawei Zhang, Yang Wang, Piero Molino, Lezhi Li, and David S Ebert. 2019a. [Manifold: A model-agnostic framework for interpretation and diagnosis of machine learning models](#). *IEEE transactions on visualization and computer graphics*, 25(1):364–373.
- Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019b. [Bertscore: Evaluating text generation with bert](#). *arXiv preprint arXiv:1904.09675*.
- Qingyu Zhou, Nan Yang, Furu Wei, Shaohan Huang, Ming Zhou, and Tiejun Zhao. 2018. [Neural document summarization by jointly learning to score and select sentences](#). In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 654–663.
- Luisa M. Zintgraf, Taco S. Cohen, Tameem Adel, and Max Welling. 2017. [Visualizing deep neural network decisions: Prediction difference analysis](#). In *Conference Track Proceedings of the 5th International Conference on Learning Representations*.