ABSTRACT

Title of dissertation: TEACHING MACHINES TO ASK USEFUL CLARIFICATION QUESTIONS

Sudha Rao
Doctor of Philosophy, 2018

Dissertation directed by: Professor Hal Daumé III
Computer Science, University of Maryland

Inquiry is fundamental to communication, and machines cannot effectively collaborate with humans unless they can ask questions. Asking questions is also a natural way for machines to express uncertainty, a task of increasing importance in an automated society. In the field of natural language processing, despite decades of work on question answering, there is relatively little work in question asking. Moreover, most of the previous work has focused on generating reading comprehension style questions which are answerable from the provided text. The goal of my dissertation work, on the other hand, is to understand how can we teach machines to ask clarification questions that point at the missing information in a text. Primarily, we focus on two scenarios where we find such question asking to be useful: (1) clarification questions on posts found in community-driven technical support forums such as StackExchange (2) clarification questions on descriptions of products in e-retail platforms such as Amazon.

In this dissertation we claim that, given large amounts of previously asked questions in various contexts (within a particular scenario), we can build machine learning models that can ask useful questions in a new unseen context (within the same scenario). In order to validate this hypothesis, we firstly create two large datasets of context paired with clarification question (and answer) for the two scenarios of technical support and e-retail by automatically extracting these information from available datadumps of StackExchange and Amazon. Given these datasets, in our first line of research, we build a machine learning model that first extracts a set of candidate clarification questions and then ranks them such that a more useful question would be higher up in the ranking. Our model is inspired by the idea of expected value of perfect information: a good question is one whose expected answer will be useful. We hypothesize that by explicitly modeling the value added by an answer to a given context, our model can learn to identify more useful questions. We evaluate our model against expert human judgments on the StackExchange dataset and demonstrate significant improvements over controlled baselines.

In our second line of research, we build a machine learning model that learns to generate a new clarification question from scratch, instead of ranking previously seen questions. We hypothesize that we can train our model to generate good clarification questions by incorporating the usefulness of an answer to the clarification question into the recent sequence-to-sequence based neural network approaches. We develop a Generative Adversarial Network (GAN) where the generator is a sequence-to-sequence model and the discriminator is a utility function that models the value of updating the context with the answer to the clarification question. We evaluate our model on our two datasets of StackExchange and Amazon, using both automatic metrics and human judgments of usefulness, specificity and relevance, showing that our approach outperforms both a retrieval-based model and ablations that exclude the utility model and the adversarial training.

We observe that our question generation model generates questions that range a wide spectrum of specificity to the given context. We argue that generating questions at a desired level of specificity (to a given context) can be useful in many scenarios. In our last line of research we, therefore, build a question generation model which given a context and a level of specificity (generic or specific), generates a question at that level of specificity. We hypothesize that by providing the level of specificity of the question to our model during training time, it can learn patterns in the question that indicate the level of specificity and use those to generate questions at a desired level of specificity. To automatically label the large number of questions in our training data with the level of specificity, we train a binary classifier which given a context and a question, predicts whether the question is specific (to the context) or generic. We demonstrate the effectiveness of our specificity-controlled question generation model by evaluating it on the Amazon dataset using human judgements.

TEACHING MACHINES TO ASK USEFUL CLARIFICATIONS QUESTIONS

by

Sudha Rao

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2018

Advisory Committee:
Professor Hal Daumé III, Chair/Advisor
Professor Philip Resnik
Professor Marine Carpuat
Professor Jordan Boyd-Graber
Professor David Jacobs
Professor Lucy Vanderwende

© Copyright by
Sudha Rao
2018

Dedication

To my loving amma (mom), appa (dad) and akka (sister).

Acknowledgments

I have been very fortunate to have got the opportunity to pursue my PhD at University of Maryland. I really enjoyed my time at UMD and I would like to thank a number of people at UMD without whom my work would not have been possible. First of, I would like to thank my wonderful advisor Hal Daumé III who guided me throughout my PhD and provided me with valuable advice. Our lengthy discussions helped me think through my ideas and motivated me to explore interesting new avenues of research. He taught me how to write papers including how to motivate the work and be thorough in explaining the technical details. I also learnt from him how to design experiments and ask the right research questions when evaluating proposed models. I am also thankful to Hal for all his career advice and for encouraging me to always aim for the best. I am also very grateful to Philip Resnik who I worked very closely with during the first two years of my PhD. His constant emphasis on thinking about the big picture helped me put my work in perspective. His course on Computational Linguistics II was my most favorite course at UMD. His enthusiasm for the material was very contagious and helped foster my interest in NLP. I am also thankful to him for guiding me through my career choices and helping me build some useful connections in the NLP community.

I am also very thankful to all my other committee members: Jordan Boyd-Graber, Marine Carpuat, David Jacobs and Lucy Vanderwende. Their valuable feedback on my thesis helped me improve my draft. A special thanks to Lucy Vanderwende for agreeing to be an external member on my committee. Her expertise in

topics closely related to my work helped me broaden my research ideas. I was very fortunate to be a part of the amazing Computational Linguistics and Information Processing (CLIP) lab at UMD. The weekly CLIP talks, reading groups, paper clinics and the opportunity to have one-on-one meetings with the speakers of the CLIP talks, have been extremely crucial to my growth as a researcher. A big thank you to all the faculty members of the CLIP lab for taking the initiative and organizing all these for the benefit of the students in the CLIP lab. I would also like to thank all the other CLIP lab faculty members: Naomi Feldman, Doug Oard, Jimmy Lin, Vanessa Frias-Martinez and Louiqa Raschid. I am also thankful to the Language Science Community for providing me with the platform to discuss my research ideas in an interdisciplinary atmosphere. Presenting my work at the Language Science Lunch Talks helped me hone the skills of describing my work to a non-NLP audience. I would also like to thank LSC for organizing the yearly Language Science Day and Winter Storm where I enjoyed meeting all the other language enthusiasts at UMD. Finally, I would also like to thank Jennifer Story, Joe Webster, Janice Perrone and Tom Hurst at UMD for making all the administrative tasks easy.

I would also like to take this opportunity to thank all my internship mentors. Thank you Daniel Marcu and Kevin Knight for providing me with my first research internship opportunity at ISI. Through this internship, I got my exposure to the new area of semantic parsing. I would also like to thank my Microsoft Research internship mentor Paul Mineiro. During this internship, I got my first hands-on experience with deep learning technology that later on turned out to be the dominant approach in my thesis. I would also like to thank my Grammarly internship mentor Joel Tetreault

for teaching me how to set concrete goals to ensure a successful research project in a short duration of an internship. Finally, I would like to give my immense thanks to Rajiv Gandhi who was the most important mentor in my undergraduate life. I would not have thought about pursuing a PhD if not for his encouragement and guidance.

My time in the CLIP lab would not have been enjoyable if not for the amazing students at the CLIP lab. Firstly, I would like to thank Allyson Ettinger, Yogarshi Vyas, Xing Niu and Trista Cao who have not only been great lab mates but also amazing collaborators. Special thanks to my senior lab members Snigdha Chaturvedi, He He and Mohit Iyyer who always inspired me to aim for more by being incredible researchers and by providing me with timely support and guidance. I also thank Hal's other students Amr Sharaf, Kiante Brantley, Khanh Nguyen and Shi Feng for providing me timely feedback on my paper drafts and presentations. I also thank all the other students in the CLIP lab who I have enjoyed having several interactions: Junhui Li, Yuening Hu, Ke Wu, Hadi Amiri, Amittai Axelrod, Ahmed Elgohary, Rashmi Sankepally, Pedro Rodriguez, Denis Peskov, Yulu Wang, Suraj Nair, Marianna Martindale, Hua He, Fenfei Gao, Viet-An Nguyen, Ke Zhai, Thang Nguyen, Ning Gao, Mossaab Bagdouri, Jinfeng Rao, Anupam Guha, Weiwei Yang, Joe Barrow, Petra Galuscakova, Jiaul Paik, Alvin Grissom II, John Morgan, Sweta Agarwal and Pranav Goel.

Last but not the least, I would like to thank my close friends and family. Thank you Meethu Malu, Manaswi Saha, Pallabi Ghosh, Nidhi Shah, Jay Ghurye and Mary DePascale for being amazing roommates. Because of you all I had a

home away from my home. I would also like to thank all my other friends at UMD: Amit Chavan, Kartik Nayak, Bhaskar Ramasubramanian, Soham De, Manish Purohit, Anshul Sawant, Zamira Daw, Ramakrishna Padmanabhan and Ladan Najafizadeh. My PhD life would not have been fun without you guys. I thank my parents immensely for believing in me and encouraging me throughout my PhD. They supported me wholeheartedly even though this meant me staying away from them for years together. I would also like to thank my dear sister and brother-in-law for supporting me and my dearest little nephew and niece for helping me keep my calm through their loaded cuteness. I also thank my in-laws for their support and love throughout my PhD. Finally, I would like to thank my loving husband Amit Rao who supported me and continues to support me accomplish my dreams and goals. Thank you all.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Motivation

An overarching goal of the natural language processing community is to develop techniques that would enable machines to process naturally occurring text as accurately as humans do. However, as humans, we may not always understand each other. In pragmatics, Grice's theory of conversational implicatures (Grice, 1975) says that there is a difference between what someone says and what someone 'implicates' by uttering a sentence. What someone says is determined by the conventional meaning of the sentence uttered and contextual processes of disambiguation; what she implicates is associated with the existence of some rational principles and maxims governing conversation. The Gricean maxims of conversation suggest speakers and listeners adhere to a Cooperative Principle where a speaker communicates information that is as informative as required and not more. The speaker assumes a certain common ground or mutual information or shared knowledge with the listener (Clark, 1981; Clark et al., 1991; Clark and Carlson, 1982). In case of gaps or mismatches in knowledge, the listener resorts to asking questions. Correction of such knowledge deficits has been identified as one of the key purposes of asking questions (Graesser et al., 2008).

With the advancements of artificial intelligence technologies, automated agents such as text-based or voice-based search engines, interactive robots, automated car navigation systems, etc are becoming increasingly common in our day-to-day lives. We frequently use these bots to search for information or accomplish certain tasks. However, often when a human user's input to a bot is underspecified i.e. it is missing some information, the bot might fail in its task. One key reasons for such failures is the lack of common understanding between the human user and the bot. The human user has a certain understanding of their problem/request and often times she fails to convey the same understanding to the bot. In such a scenario, the bot can be much more useful if it could try to establish this common understanding by asking relevant questions. For example, if we search in a search engine "How long does it take to get a PhD", then the search engine could in turn ask "In which field?" since the duration of the program would differ according to the field of study. Or if we instruct a robot "Please bring me my coffee mug from the kitchen" and if there are multiple mugs in the kitchen, the robot could in turn ask "What color is your coffee mug?" in order to distinguish our mug from the other mugs in the kitchen. If we wish to make such human-bot interactions as efficient as human-human interactions are, it is important that we teach machines to ask clarification questions when faced with uncertainty or knowledge gaps. We define "clarification question" as a question that asks for some missing information in a given context.

In the field of natural language processing, however, despite decades of work on question answering, there has been little work in question asking. Moreover most of the previous work on generating questions has been on generating reading

comprehension style questions: given a text, write a question that one might find on a standardized test with the goal of assessing someone's understanding of the text. Comprehension questions, by definition, are answerable from the provided text. Clarification questions, on the other hand, ask about information that is missing from the given text and hence is not answerable from the text. The goal of my thesis work is to explore how can a machine automatically generate clarification questions when faced with uncertainty or knowledge gaps. More concretely, we define our goal as given a context, we want to automatically generate a question whose answer can fill in the information missing from the given context.

1.2 Specific Scenarios Considered in this Dissertation

Text generation has been studied extensively in the field of natural language processing. Tasks such as machine translation, summarization, dialogue generation have achieved varied degrees of success in this field. Most of the successful models have been machine learning models where the models learn from vast amounts of data. For instance, in the task of machine translation, models learn to translate from say French to English by having access to large number of French-English sentence pairs where the French sentence has been translated into English or vice-versa. The French sentence in this case can be considered as the input whereas the English sentence can be considered as the label or the output. A supervised machine learning model will then learn to predict the label (or the output) given the input. Motivated by these successes, in this thesis, we take a supervised learning

approach for our task of generating useful clarification questions given a context. Supervised learning approaches need access to large amounts of labelled data or large amounts of input-output pairs (in our setting). We, therefore, approach the problem of clarification question generation under two specific scenarios where we have access to abundant such input-output data online.

Our first scenario is the generation of clarification questions during troubleshooting of technical issues. About a decade ago, if you faced a technical problem the only way to solve it would be to go to an expert. Due to the recent surge in the use of internet, most of the problem solving these days happen online on question answering (Q&A) forums where users post their problems and others provide assistance by replying to the posts. However, Asaduzzaman et al. (2013) observed that on StackExchange, which is one such community-driven problem solving platforms, posts often go unanswered for a long time because they are not clear enough i.e. they are missing some information. Consequently, other users ask clarification questions to those posts so that they can better offer assistance to the original poster. For instance, in Figure 1.1, a user posts an issue she is facing while installing a certain software on Ubuntu operating system. Another user on the forum asks for the version of Ubuntu in the comment section of the post suggesting that the version information could be useful in debugging the issue and hence should have been included in the initial post. In this dissertation, we train a machine learning model that learns to automatically generate a useful clarification question given an underspecified post. We imagine a use case in which while a user is writing their post, a system generates a single (or a shortlist of) question(s) asking for information that

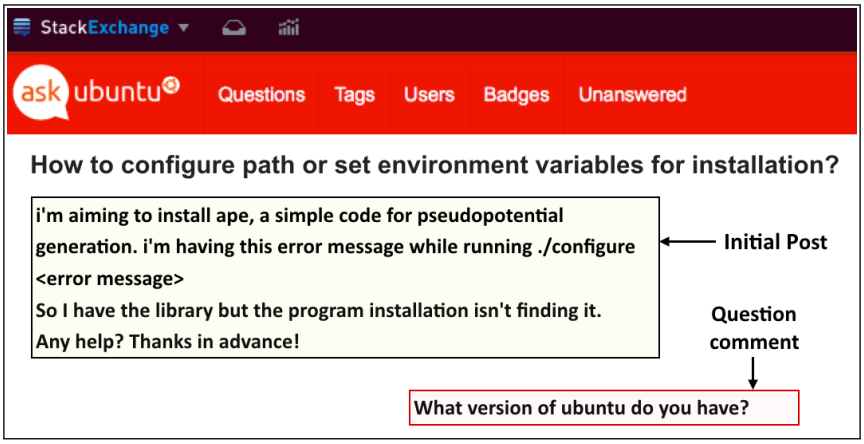

Figure 1.1: Sample post paired with a clarification question on StackExchange, an online question-answering forum.

it thinks other users on the forum might need to provide a solution, thus enabling the original poster to immediately clarify their post, potentially leading to a much quicker resolution.

Our second scenario is the generation of clarification questions during online shopping on e-retail platforms such as amazon.com. With the emergence of internet, people frequently resort to online shopping for buying different products. Often times the description of a product on these e-retail platforms could omit important information that a potential buyer might seek. For example, Figure 1.2 shows the description of a cookware set on amazon.com. A potential buyer asks "Are they ok for induction stove?" in the FAQ section pointing out that this information about the compatibility of the pan with induction stove tops is missing from the current product description. In this dissertation, we train a machine learning model that learns to automatically generate a useful clarification question given a product description. As in the previous scenario, we imagine a use case in which while a

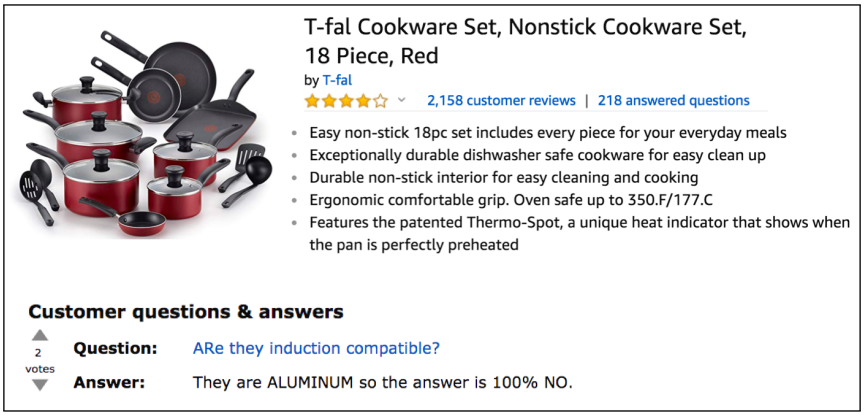

Figure 1.2: Sample product description paired with a clarification question on Amazon, an online shopping platform.

product seller is writing their initial product description, a system generates a single (or a shortlist of) question(s) asking for information that it thinks a potential buyer might need to make a more informed decision about the purchasing of the product, thus enabling the seller to immediately clarify their description. In future, one could also imagine building systems that can in turn answer these auto-generated questions from other similar product descriptions or product reviews.

1.3 Contributions

Our first contribution is the creation of two clarification questions dataset. Most previous work on question generation took the approach of transforming statements into questions using syntactic rules. We instead take a novel approach where we investigate how can we use existing human written questions to train a machine learning model to in turn generate new questions. We hypothesize that given abundant amounts of such naturally occurring questions, we can build a machine learning

model that can learn useful patterns of question asking and generalize those to new unseen contexts. We build our first dataset for the scenario of technical support. We use existing StackExchange datadump to find posts on which people have asked clarification questions and the author of the post has subsequently answered the question to create our dataset of (context, question, answer) triples. We build our second dataset for the scenario of online shopping (e-retail). We use existing Amazon dataset to find product descriptions on which people have asked clarification questions and the product seller (or another user) has answered the question to create our dataset of (context, question, answer) triples.

In both the aforementioned scenarios, similar contexts tend to reoccur frequently. For instance, under the StackExchange scenario, a post describing the issue with the installation of a certain software X on Ubuntu operating system might have similarities with another post describing the issue with the installation of a different software Y on Ubuntu. Therefore, a question such as "What version of Ubuntu are you using?" previously asked on a certain post could be useful for a new post as well. Similarly, under the Amazon scenario, a kitchen appliance such as toaster might share common features with another appliance such as a sandwich maker. Therefore, a question such as "How long is the cord?" asked about toaster, can be a useful question about sandwich maker as well. This motivates a learning approach that looks at questions asked previously to contexts that are similar to the given context and chooses a question from that candidate set that could be useful to the given context as well. Our second contribution is a novel question ranking model which first extracts a set of candidate questions from a pool of previously

asked questions based on the similarity with the given context and then ranks these questions in a way that a question more useful to the given context would be higher up in the ranking.

A major limitation of the ranking approach is that it can only reuse the questions already existing in a dataset. It cannot generalize to new unseen scenarios. For example, under the StackExchange scenario, if the previous posts only discuss issues faced when using Ubuntu operating system, the model will not be able to generate a question such as "What version of Windows are you using?". We hypothesize that we can train a sequence-to-sequence learning model (Sutskever et al., 2014) to generate a clarification question one word at a time, given the context as the input. Our third contribution is a clarification question generation model trained to maximize an answer-based utility function. We use an approach similar to the more recent generative adversarial networks (Goodfellow et al., 2014) to train our model.

We observe that humans ask clarification questions at different levels of specificity. For instance, in Figure 1.1, the question "What version of Ubuntu are you using?" is a generic question i.e. it could be useful for many other posts. Whereas, a question such as "Does your bashrc file include the path to the library installation?" is specific to the given post. In Figure 1.2, the question "Are they ok for induction stove?" is a question specific to the given product whereas the question "Is there a guarantee or warranty?" is a generic question. We hypothesize that we can guide a machine learning model to generate questions at a desired level of specificity by providing the level of specificity as a signal while training the model.

Our fourth contribution is a specificity-controlled question generation model which given a context and a level of specificity (specific or generic), generates a question at that level of specificity.

1.4 Roadmap

Chapter 2 presents a background study where we first define "clarification questions" in more detail and discuss their importance. We describe some of the existing works on question generation in the natural language processing literature. We also review the different approaches to question generation including the traditional syntax based methods and the more recent neural network based methods. The chapter also includes a brief review of the neural network models most relevant to this dissertation. Finally, related to our specificity-controlled question generation model, we discuss some of the recent works on generating text controlled for a given style.

1.4.1 Dataset Creation

Chapter 3 describes our method for creating the StackExchange and the Amazon clarification questions dataset. We begin by discussing the importance of asking clarification questions in these two scenarios. We then describe in detail how we extract the (context, question, answer) triple from the raw data including the preprocessing steps. To create the dataset for the StackExchange scenario, we use the

publicly available StackExchange datadump. On StackExchange, users routinely ask clarification questions to post. The author of the post subsequently edits the post answering the question. We use the StackExchange's edit history to extract the initial post as the "context", the question asked in the comment section of the post as the "question" and the edit made to the post in response to the question as the "answer" to create our dataset of (context, question, answer) triples. To create the dataset for the Amazon scenario, we repurpose the formally created Amazon product review dataset (McAuley et al., 2015) and the Amazon question-answering dataset (McAuley and Yang, 2016). We extract the product description as the "context", the question asked by a potential buyer in the FAQ section of the corresponding product as the "question" and the response given by the seller (or an existing customer) to the question as the "answer" to create our dataset of (context, question, answer) triples. We also include some data analysis.

1.4.2 Question Ranking Model

Chapter 4 introduces our novel question ranking model. In our learning model, we represent the words in the context, question and answer using word embeddings (Mikolov et al., 2013; Pennington et al., 2014) which correspond to vector representations of words in some N-dimensional space in a way that words that are closer in meaning would be closer in the vector space. From these word level representations, we obtain the sentence level representation using recurrent neural networks (Hochre-

iter and Schmidhuber, 1997; Mikolov, 2010) which perform a series of non-linear transformations on the input word vectors guided by a task specific loss function. Such neural network models have recently proven to be effective for several natural language processing tasks such as part-of-speech tagging (Santos and Zadrozny, 2014), dependency parsing (Chen and Manning, 2014), sentiment analysis (Glorot et al., 2011), etc. Our neural network model is inspired by the decision theoretic framework of expected value of perfect information (EVPI). EVPI is a measurement of the value of gathering information. We use EVPI to calculate which question is most likely to elicit an answer that would make the post more informative. Given a context and a set of candidate questions, we rank the questions by their EVPI value.

In this chapter, we start by describing the notion of Expected Value of Perfect Information (EVPI) and then discuss how we model our problem under the EVPI framework. We describe the three components of our model: question & answer generator, answer model and utility calculator, and describe the details of our neural network based representations. We discuss our human-based evaluation design and conclude with the results of our experiments on the StackExchange dataset.

1.4.3 Question Generation Model

Chapter 5 introduces our novel question generation model. Our question generation model is built on the sequence-to-sequence approach that has proven effective

for several language generation tasks (Du et al., 2017; Serban et al., 2016b; Sutskever et al., 2014; Yin et al., 2016). Unfortunately, training a sequence-to-sequence model directly on context/question pairs yields generated questions that are highly generic. For instance, in the context of asking questions about home appliances, these models frequently generate bland questions such "What are the dimensions?" or "Is it made in China?',' corroborating a common finding in dialog systems (Li et al., 2016b). Our goal is to be able to generate questions that are useful *and* specific. Inspired by the idea of expected value of perfect information, we build a model that uses the answer to the generated question to decide the usefulness of the question by measuring the value of updating the context with the answer. We construct a model that first generates a question given a context, and then generates a hypothetical answer to that question. Given this (context, question, answer) tuple, we train a utility calculator to estimate the usefulness of this question. We reinterpret the utility value as a reward in reinforcement learning setting and train our model to generate questions that will give us a high reward.

Reinforcement learning is one of the learning paradigms under machine learning which unlike supervised learning does not assume access to input-output pairs apriori. Given some input, the model makes predictions and gets a reward for that prediction from the environment. The goal of the model is to maximize its end reward. The use of such a reward based learning strategy relaxes the strong dependence on input-output pairs that is otherwise observed in a supervised learning strategy. This is especially attractive in our problem setting since we find that a given context can have multiple useful clarification questions. For instance, in Fig-

ure 1.2, a question such as "Is there a guarantee or warranty?" could be useful as well even though the question is very different from the one that the product was paired with in the dataset. The use of a reward based learning strategy allows the model to quantify the usefulness of a generated question by its utility value as opposed to its similarity to the question paired with the given context in the dataset.

Further, we improve the utility calculator by training it along with our question generation model. We show that the utility calculator can be generalized using ideas for generative adversarial networks (Goodfellow et al., 2014) for text (Yu et al., 2017). A generative adversarial network is a training procedure for "generative" models that can be interpreted as a game between a generator and a discriminator. The goal of the generator is to generate data such that it can fool the discriminator; the goal of the discriminator is to be able to successfully distinguish between real and generated data. In the process of trying to fool the discriminator, the generator produces data that is as close as possible to the real data distribution. In our problem setting, the utility predictor plays the role of the "discriminator" and the question generator is the "generator" and we train our model end-to-end using adversarial training approach. We find that our adversarially-trained model generates questions that are more specific to the context.

In this chapter, we begin by describing the sequence-to-sequence neural network framework on which we base our question generation model. We then discuss the limitations of such sequence-to-sequence models and motivate the use of utility-function based reward. Further, we describe the generative adversarial training paradigm and discuss how we reinterpret our utility predictor in this adversarial

training setting. Finally, we describe our automatic metric-based and human-based evaluation strategy and present results of our experiments on both the StackExchange and the Amazon dataset.

1.4.4 Specificity-Controlled Question Generation Model

Chapter 6 introduces our specificity-controlled question generation model. There has been previous work on generating text with specific stylistic constraints both at the lexical (Edmonds and Hirst, 2002; Inkpen and Hirst, 2006; Kamps et al., 2004; Reiter et al., 2005) and more recently at sentence level (Jhamtani et al., 2017; Xu et al., 2012; **?**). Our model is primarily based on the idea of side constraints where the source is appended with an artificial token denoting the style in which we want the model to generate its target. This idea has been used before for controlling politeness (Sennrich et al., 2016), voice (Yamagishi et al., 2016), and formality (Niu et al., 2017, 2018) in machine translation. In our setting, the side constraint corresponds to the level of specificity. We annotate a set of 3000 questions from the Amazon dataset with their level of specificity using crowdsourcing. Next, we train a model to automatically identify the level of specificity given the context and the question (Louis and Nenkova, 2011). We use this model to append the source context with the level of specificity of the target question. We finally retrain our previously described question generation model with the modified source. At test time, given a context and a level of specificity, our model generates a clarification question at that level of specificity.

In this chapter, we begin by describing our method for annotating the questions with their level of specificity using crowdsourcing. We then describe our feature based learning model for automatically identifying the level of specificity of clarification questions. Finally, we describe how we integrate these specificity annotations as side constraints into our question generation model. We present the results of our experiments on the Amazon dataset.

1.4.5 Future Directions

Chapter 7 concludes this thesis by summarizing our contributions and discussing the shortcomings of our approaches. We also present some avenues of future work where teaching machines to ask useful questions would be helpful. For instance, in the context of goal oriented dialogue, teaching an agent to ask the right questions to a human can help the agent successfully solve a task. In the context of writing assistance, teaching machines to identify important gaps in the content and ask the right questions to the human writer can help the writer fill those informational gaps. Another potential direction is the use of multi-modal inputs to guide machines to ask useful questions. For instance, in the context of robot navigation, teaching a robot to ask the right questions using both visual context (surrounding environment) and textual context (human interaction) can help the robot resolve its uncertainty and thus enable it to navigate more easily in a given environment. The skill of asking the right questions is an important yardstick of human intelligence and therefore teaching machines to ask useful questions can take us a step closer to

building intelligent artificial agents.

Chapter 2: Background

2.1 Definition and Importance of Clarification Questions

We define "clarification question" as a question that asks about some information "X" that is currently missing from a given context but is essential for someone trying to solve a task or make a decision using the given context. Graesser et al. (1992) identify the four different purposes of questions as correction of knowledge deficits (e.g. information seeking question such as *"What is the color of the coffee mug?"*), monitoring common ground (e.g. *"Are we meeting after lunch today?"*), social coordination of action (e.g. *"Can you please close the door behind you?"*) and control of conversation and attention (e.g. *"Hello Sir, how are you doing today?"*). Our definition of clarification question aligns most with the first purpose: correction of knowledge deficits. We consider this definition of clarification questions in the context of problem solving. This definition is subsumed by the broader definition of clarification questions which includes any questions whose purpose is to eliminate confusion, ambiguity or misunderstanding and seek additional essential information. Clarification questions can be of two types: open questions and closed questions. Open clarifying questions are the ones that take the form of what, when, where, which, how and why questions (e.g. *How are you installing this soft-*

ware?). Whereas, closed clarifying questions take the form of yes/no questions (e.g. *Do you have Powerpoint installed in your computer?*). Clarification questions are sometimes also known as probing questions since they probe the participants of the discussion to give more information on what they said.

Asking questions is considered to be central to learning, cognition and education. Researchers in education and development psychology have found that teaching students to ask probing questions in a classroom setting can help foster their learning (Graesser and Person, 1994; Rosenshine et al., 1996). The art of asking good probing questions requires a deep understanding of the subject matter and the ability to identify what is the essential missing information. Therefore, learning to ask useful questions can help students develop essential skills such as reasoning, problem solving and knowledge building. Adults ask clarification questions often when participating in discussions. Asking clarification questions helps the speaker and the listener establish a common ground which is required for an effective communication.

With the advancements of artificial intelligence technologies, we find ourselves interacting more and more with automated agents in our daily lives. In order for these agents to be successfully communicating with humans, it is important that they are able to establish the same mutual common ground with humans. Therefore, it is important that these agents learn how to ask clarification questions to humans when they face with uncertainty. Learning to ask useful questions would also help these agents achieve the same kind of reasoning and understanding abilities as humans (Vanderwende, 2008).

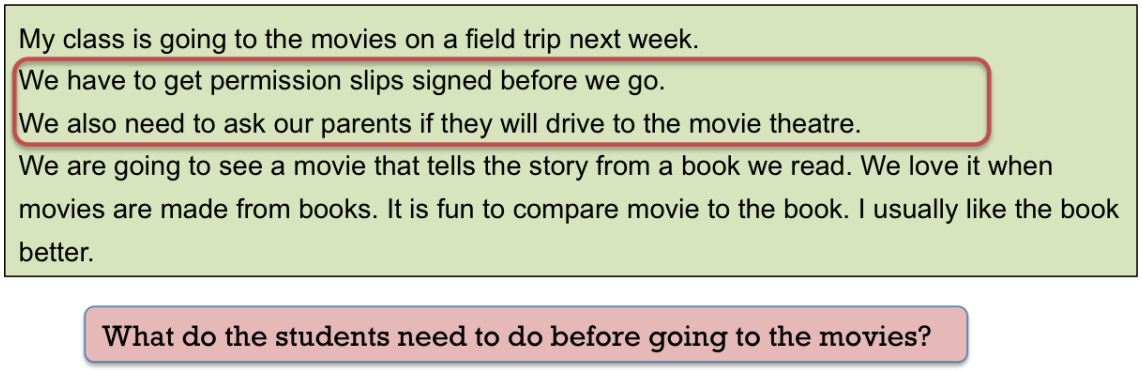

Figure 2.1: An example of a reading comprehension passage and a question whose answer can be found in the given passage from Heilman (2011).

2.2 Different Types of Question Generation Work

2.2.1 Reading Comprehension Question Generation

The task of question generation is defined as automatically generating a question, given a context. Most previous work on question generation has been on generating reading comprehension style questions: given text, generate a question whose answer can be found in the given text (Heilman, 2011; Olney et al., 2012; Rus et al., 2011; Vanderwende, 2008). For instance, in Figure 2.1, given the passage highlighted in green, the task is to generate a question such as the one highlighted in red with the goal of assessing someone's understanding of the given passage. Automatically generating such reading comprehension questions can be helpful in creating standardized tests. In this dissertation, on the other hand, our goal is to generate questions whose answer cannot be found in the given text. Therefore, the challenge in our work is not limited to identifying relevant information from a

given text, but requires a broader understanding of the subject at hand and asking questions that can identify important missing information in a given text.

2.2.2 Question Generation in Dialogue

Outside reading comprehension, the task of question generation has been studied the most under the context of task oriented dialogue (Grosz and Sidner, 1986). In a task oriented dialogue, a system interacts with a human with the purpose of accomplishing a given task. For instance, consider the task of flight booking. A system interacting with a human user would ask the user a set of questions that would enable the system to book a flight for the human user. Under the context of such task oriented dialogue, the intent of teaching a system to ask questions is to fill some predefined slots (Bobrow et al., 1977; Goddeau et al., 1996; Lemon et al., 2006; Williams et al., 2013; Young et al., 2013). For instance, in travel booking, the slots would include origin city, origin time, airline etc (refer Figure 2.2). Correspondingly, the system would generate questions such as *"What time do you want to leave?"*, *"Which airline would you prefer?"*, etc that would help fill those predefined slots. In contrast, in our work we do not assume access to such predefined slots apriori. The goal is to identify these missing slots implicitly and ask a question about it.

Another use case of question generation in dialogue is to resolve ambiguity. For instance, in spoken dialogue, due to error prone automatic speech recognition (ASR) systems, clarifying the intent of the user becomes important. Clark (1996) and Allwood (2000) argue that the aim of clarification questions in human-human

USER:	I want to go to **Melbourne** on **July 14**
SYSTEM:	What **time** do you want to leave?
USER:	I must be in Melbourne by 11 am
SYSTEM:	Would you like a **Delta** flight that arrives at 10.15 am?
USER:	Sure
SYSTEM:	In what **name** should I make the reservation?

SLOTS

<origin city>
<departure city>
<origin time>
<departure time>
<airline>

Figure 2.2: An example interaction between a user and a system in the context of travel booking where they system asks questions to fill a set of predefined slots.

dialogue is to resolve misunderstanding at the following levels: securing attention, hearing an utterance, meaning of an utterance and deciding which action is appropriate. Most spoken dialogue systems ask generic clarification questions such as *"What did you say?"* or *"Can you please repeat?"* when faced with uncertainty. Stoyanchev et al. (2014), on the other hand, develop a model with the aim of generating more targeted clarification questions. For instance, consider the interaction below:

A: When did the problems with [*power*] start?

B: The problem with what?

A: Power

Speaker B asks a targeted clarification question instead of merely saying *"Please repeat"*. They present an approach for generating more natural clarification questions using rules based on human behavior. Our work is similar to this work in that our goal is also to generate more natural clarification questions. However, in our work, the ambiguity in the original intent is not because of failure of the ASR system but because of a piece of information that is missing from the given context. Hence, we

aim to generate clarification questions that point at the missing information instead of those that point at an information that is unclear in the given context.

Following Stoyanchev et al. (2014), there have been other similar works such as recognizing intention through clarification dialogue (Trott et al., 2016) and entity disambiguation through clarification dialogue (Coden et al., 2015). Our work is similar to these in that we also aim to generate questions to better understand the original intent. But our work aims to resolve such ambiguity at a more general level by asking for missing information instead of specifically disambiguating an intent or an entity.

2.2.3 Question Generation for Text Understanding

Liu et al. (2010) propose a novel question generation model that generates trigger questions as a form of support for students' learning through writing. For instance, if a student is writing a related work section, then their system would generate questions that would help the student augment their writing with supporting arguments. Figure 2.3 shows an example use case of this system where, when the author writes an argument, the system generates the questions that encourage the author to provide more reasoning to the argument. Our work is similar to this work in that our aim is also to augment the given context with additional informational content. However, the intent of generating questions in our scenario is to resolve an uncertainty or to fill informational gaps rather than help the author improve their understanding of the original text.

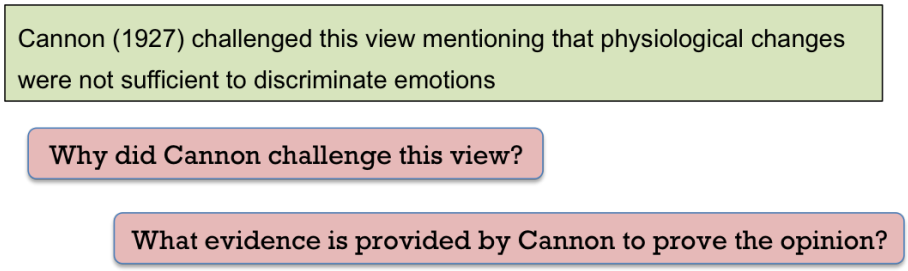

Figure 2.3: An example of an argument followed by questions that encourage further explanation of the argument from Liu et al. (2010)

SYSTEM:	How can I help you?
USER:	I would like to fly from Atlanta Georgia to London England on September 24th in the early evening. I would like to return on October 1st departing from London in the late morning.
SYSTEM:	Leaving what city?
USER:	Atlanta Georgia
SYSTEM:	Going to which city?
USER:	London
[conversation continues]	

Figure 2.4: Example of a system asking series of questions to simplify the original user query from Artzi and Zettlemoyer (2011)

Artzi and Zettlemoyer (2011) use human-generated clarification questions to drive a semantic parser where the clarification questions are aimed towards simplifying a user query. For instance, consider the user query shown in Figure 2.4. In order to simplify the complex query, the system in turn asks the user follow-up questions that helps the system parse the original user query more easily. Our work departs from this work in that we generate questions to fill some missing information in a given text instead of generating questions that reiterates something that was already stated before.

2.2.4 Visual Question Generation

Most previous work at the intersection of language and image processing has been in caption generation where given an image the goal is to generate a caption that explains the image. For instance, given an image such as the top image in Figure 2.5, the goal would be to generate a caption such as "A man and a woman standing next to a fallen motorcycle". However, recently Mostafazadeh et al. (2016) introduced the visual question generation task where the goal is to generate natural and engaging questions about an image. For instance, for the top image in Figure 2.5, the goal is to generate questions such as "Was anyone injured?", similar to what a human would think about when they look at this image. Somewhat similar to clarification questions in our work, these questions do not ask about something that is already present in the image but rather ask about something that can be inferred from the given image. Following this work, Mostafazadeh et al. (2017) introduced an extension of this task called the Image Grounded Conversation task where they use both the image and some initial textual context to generate a natural follow-up question and a response to that question. Our work departs from these work in that, given a context, we assume there is a goal to be accomplished using the given information (which is more specific than say the broader goal of image understanding suggested perhaps by Mostafazadeh et al. (2016)). And therefore, the questions generated by our work aim at asking for information that can help someone achieve that goal faster.

Q: Was anyone injured in the crash?

Q: Is the motorcyclist alive?

Q: What caused the accident?

User1: My son is ahead and surprised!

User2: Did he end up winning the race?

User1: Yes he won, he can't believe it!

Figure 2.5: Example from the Visual Question Generation task (Mostafazadeh et al., 2016) and the Image Grounded Conversation task (Mostafazadeh et al., 2017).

2.2.5 Question Refinement to Help Question-Answering

The task of question-answering can be defined as given a question, retrieve (or generate) an answer to the question from a document (or a set of documents) or a database. Previous work in question-answering find that retrieving the correct answer could largely depend on the way the question is asked. Therefore, there has been work on refining a given question with the aim of improving the accuracy of a question-answering system. For instance, the keywords to questions (K2Q) system (Zheng et al., 2011) generates a list of candidate questions and refinement words, given a set of input keywords, to help a user ask a better question. Figueroa and Neumann (2013) rank different paraphrases of query for effective search on forums.

Romeo et al. (2016) develop a neural network based model for ranking questions on forums with the intent of retrieving similar other question. Buck et al. (2017) propose an active question answering model where they build an agent that learns to reformulate the question to be asked to a question-answering system so as to elicit the best possible answers. As a future direction, one could imagine building a complementary system to our work which can automatically answer the questions generated by our system with the help of previous related contextual information.

2.3 Approaches to Question Generation

In this section, we describe the different major approaches to question generation explored by the natural language processing community.

2.3.1 Syntactic Rule based Methods

Given that most previous work on question generation has been on reading comprehension style question generation, the task of question generation then turns out to be, given a sentence (or a text), transform the sentence to a question. For instance, given a statement *"John met Sally"*, their system generates *"Who met Sally?"*, *"Who did John meet?"* and *"Did John meet Sally?"*. One way to achieve this would to identify name entities or adjunct roles in the statement and map them to the appropriate Wh-question. For instance, the sentence *"<Person>Albert Einstein</Person> developed the theory of relativity."* may be transformed into the question *"Who developed the theory of relativity?"* by mapping the Person named

entity "Albert Einstein" to the question type "Who". Enumerating rules for such wh-movement based transformations can be sometimes challenging, especially in the English language. For instance, the sentence *"James Madison, following Thomas Jefferson, was elected as the 4th president of United States."* should be transformed into *"Following Thomas Jefferson, who was elected as the 4th president of United States?"* instead of the awkward transformation *"Who, following Thomas Jefferson, was elected as the 4th president of United States?"*

Heilman (2011) propose a three step approach for factoid question generation where they first extract a set of factual statements from complex input texts, transform the factual statements into candidate questions and then rank them such that a better question is higher up in the ranking. Their system uses semantic entailment and presupposition for the extraction of sets of simplified factual statements from embedded constructions in complex input sentences, Given the simplified statements, they identify the answer phrases that may be targets for WH-movement and convert them into question phrases. Lastly, they use a feature-based linear regression model to rank the candidate questions.

Rus et al. (2011, 2010) introduced the question generation shared task where the task is defined as generate a question from a paragraph and generate question from a sentence such that the answer to the question can be found in the corresponding paragraph or the sentence. The systems submitted to these tasks mainly used handcrafted rules and features for generating questions (Ali et al., 2010; Kalady et al., 2010). Under template based methods, Chen (2009) generate questions from knowledge structure by filling templates *"Why/How did <character>*

<verb> <complement>?". Olney et al. (2012) generate questions from knowledge representation modeled as a concept map. Labutov et al. (2015) generate high-level question templates by crowdsourcing and given a text segment, rank question templates that are relevant. However the crowdsourcing method of collecting data leads to significantly less data than we collect using our method.

2.3.2 Neural Network based Methods

Sequence-to-sequence based network based models (explained in detail in §2.4) have had significant success at a variety of text generation tasks, including machine translation (Bahdanau et al., 2015; Luong et al., 2015), summarization (Nallapati et al., 2016), dialog (Bordes and Weston, 2016; Li et al., 2016a; Serban et al., 2016b, 2017), textual style transfer (Jhamtani et al., 2017; Kabbara and Cheung, 2016; Rao and Tetreault, 2018) and question answering (Serban et al., 2016b; Yin et al., 2016). The key idea behind these sequence-to-sequence approaches is that given large amounts of input, output sequence pairs, the model learns internal representations such that at test time, given an input sequence, it generates the appropriate output sequence.

Recently, there have been work on generating reading comprehension style questions using such neural network models. Serban et al. (2016a) created a large (30 million) factoid question-answering dataset by transforming facts in the Freebase into natural language questions. Their question generation model was inspired by the well-known attention-based encoder-decoder model (Luong et al., 2015) used for

machine translation. Duan et al. (2017) extract a large number of question-answer pairs from community question answering forums and use them to train an attention-based sequence-to-sequence learning approach to generate challenging questions for the reading comprehension task. They find that their approach outperforms previous rule-based question generation approaches when evaluated using automatic metrics and human judgments. Du et al. (2017) propose an attention-based encoder-decoder model for generating questions from text passages and show that humans find the questions generated by their model to be more natural and more difficult to answer compared to rule-based systems.

There has been also work on using neural networks for building question generation models that in turn assist question answering. Yuan et al. (2017) use sequence-to-sequence learning approach to generate natural language questions from documents, conditioned on answers. Their question generation model maximizes a reward which is defined by the performance on a downstream question answering task. Sachan and Xing (2018) propose a self-training method for jointly learning to ask and answer questions. Their model is also based on sequence-to-sequence learning with soft attention. Tang et al. (2018) use generative adversarial network (GAN) based approach for jointly learning the tasks of question answering and question generation.

Under visual question generation, Mostafazadeh et al. (2016) propose a neural network based approach for question generation where they process the image input using a convolutional neural network (CNN) and the text input using a recurrent neural network (RNN). Li et al. (2018) propose to jointly train the two tasks of visual

question generation and visual question answering using recurrent neural networks.

In our work, we use sequence-to-sequence based neural network to generate clarification question, given a textual context. In Chapter 5, we describe our question generation model where we begin with a maximum-likelihood based training approach followed by a reinforcement learning based training. Our final model uses a generative adversarial training approach to train a sequence-to-sequence based neural network model.

2.4 Relevant Neural Network Models

Applying machine learning algorithms to natural language data requires transforming text into numeric representation as a first step. Until recently, the dominant approach to learning such representations has been the use of hand-crafted features that are developed based on the task at hand. Deep learning or neural network modeling (Goodfellow et al., 2016) allows us to automatically learn representations of text without requiring feature engineering. In this section, we give an overview of the neural network models used in this dissertation.

2.4.1 Feedforward neural network

Feedforward neural networks or multilayer perceptrons are functions that perform a series of nonlinear transformations on a given input vector to obtain an output. The term feedforward comes from the fact that in these models, information flows from the input into intermediate computations and finally to the output.

There are no feedback connections in which the output is fed back into the model. In our work, we use a feedforward neural network to compute a value between 0 and 1, given an input vector. In Figure 2.6, the input layer consists of the input vector ($\vec{x} = \{x_1, x_2, ..., x_n\}$) of n dimensions, the hidden layers consists of hidden units (h_i) and the output layer consists of single output unit y. We use a fully connected feedforward neural network consisting of K hidden layers where each hidden unit h_i in the input layer l_k is connected to each of the units in the next hidden layer l_{k+1}. Each of the connections correspond to a nonlinear transformation such as a *tanh*.

$$h_i^k = \tanh(\vec{w_i^k} \cdot \vec{o^{k-1}} + b_i^k) \tag{2.1}$$

$$o_i^k = \tanh(h_i^k) \tag{2.2}$$

$$y = \text{sigmoid}(o_1^K) \tag{2.3}$$

$\vec{w_i^k} = \{w_1^k, w_2^k, ..., w_{r^k}^k\}$

$\vec{o^k} = \{o_1^k, o_2^k, ..., o_{r^k}^k\}$

w_{ij}^k : weight for hidden unit h_j^k in layer l_k for incoming hidden unit h_i^{k-1} in layer l_{k-1}

b_i^k : bias for hidden unit i in layer l_k

h_i^k : hidden unit i in layer l_k

o_i^k : output for hidden unit i in layer l_k

r^k : number of hidden units in layer l_k

This network is trained (i.e. the weights w and the bias b are learned) using

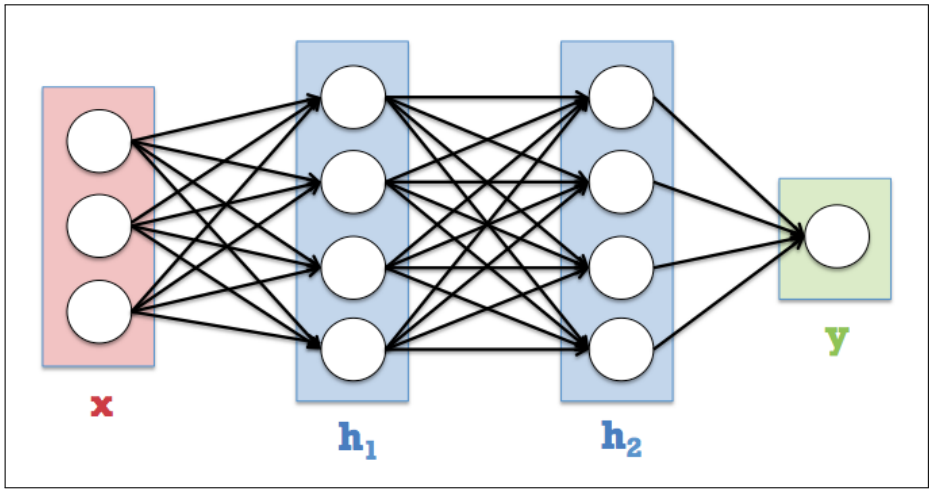

Figure 2.6: A fully connected feedforward neural network with an input layer, two hidden layers (with four hidden units each) and an output layer.

backpropagation to minimize loss such as the cross-entropy loss between all (x, y) pairs in the training data.

2.4.2 Recurrent neural network

The feedforward neural network described before cannot make use of sequential information present in language. It processes each word in the sentence independently without considering the dependencies between those words. However, in language, words in a sentence are related to each other. For instance, to predict the next word in a sentence, we need to look at the previous words. Recurrent neural networks (RNN) allows us to capture these dependencies (Hopfield, 1982; LeCun et al., 1990). Given an input sequence $(x_1, x_2, x_3, ..., x_n)$, RNN reads the input from left to right and computes an hidden state h_t at each timestep t. The hidden state is computed using both the input at the current timestep x_t and the hidden state

from the previous timestep h_{t-1}.

$$h_t = \sigma_h(W_h x_t + U_h h_{t-1} + b_h) \tag{2.4}$$

$$o_t = \sigma_y(W_y h_t + b_y) \tag{2.5}$$

where x_t: input vector

h_t: hidden layer vector

y_t: output vector

W_h, U_h, W_y: weight matrices (parameters)

b_h, b_y: bias (parameter)

σ_h, σ_y: nonlinear activation functions such as $tanh$

The model is trained using a variant of backpropagation called *backpropagation through time*. RNNs suffer from the issue of vanishing gradient when the input sequences are very long. Long-short term memory (LSTM) (Hochreiter and Schmidhuber, 1997) networks are a variant of RNNs that try to overcome this issue by having an extended memory which allows them to remember inputs over a long period of time.

2.4.3 Sequence-to-sequence neural network

We describe sequence-to-sequence learning model (Sutskever et al., 2014). Given an input sequence $x = (x_1, x_2, ..., x_N)$, this model generates an output sequence $y = (y_1, y_2, ..., y_T)$. The architecture of this model is an encoder-decoder

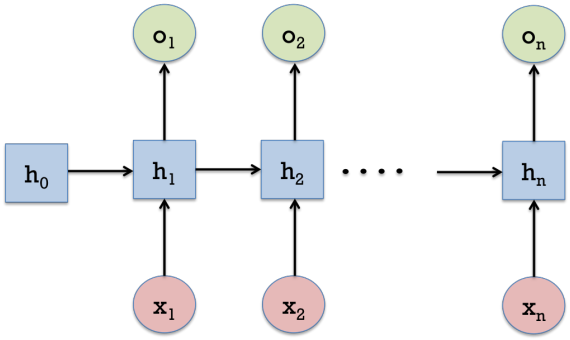

Figure 2.7: Recurrent neural network operating over the input sequence x one word at a time.

with attention. The encoder is a recurrent neural network (RNN) operating over the input word embeddings to compute a source representation $\tilde{S}$. The decoder uses this source representation to generate the target sequence one word at a time:

$$p(y|S) = p(y_1, y_2, ..., y_T|S) = \prod_{t=1}^{T} p(y_t|y_1, y_2, ..., y_{t-1}, S) \tag{2.6}$$

In the above equation, the chain rule permits the calculation of the joint distribution of the output token probabilities using the product of the individual output token probabilities. The predicted token y_t is the token in the vocabulary that is assigned the highest probability using a softmax function. The standard training objective for sequence-to-sequence model is to maximize the log-likelihood of all (x, y) pairs in the training data D.

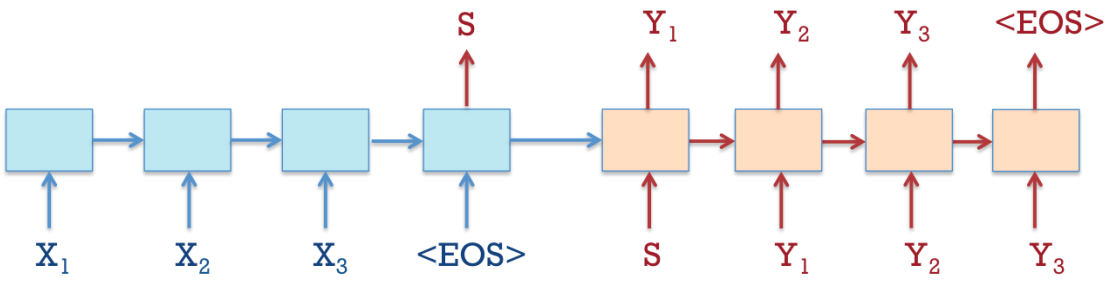

Figure 2.8: Sequence-to-sequence learning model which takes in an input sequence and generates an output sequence one word at a time.

2.5 Generating Text with Stylistic Variations

In the field of natural language processing, the task of automatically generating text in a particular style has been studied using parallel data and without using parallel data. Under style transfer using parallel data, Sheikha and Inkpen (2011) collect pairs of formal and informal words and phrases from different sources and use a natural language generation system to generate informal and formal texts by replacing lexical items based on user preferences. Xu et al. (2012) was one of the first works to treat style transfer as a sequence to sequence task. They generate a parallel corpus of 30K sentence pairs by scraping the modern translations of Shakespeare plays and train a phrase-based machine translation system to translate from modern English to Shakespearean English. More recently, Jhamtani et al. (2017) show that a copy-mechanism enriched sequence-to-sequence neural model outperforms Xu et al. (2012) on the same set. In text simplification, the availability of parallel data extracted from English Wikipedia and Simple Wikipedia (Zhu et al., 2010) led to the application of phrase-based machine translation (Wubben et al., 2012) and more

recently neural network based machine translation (Wang et al., 2016) models.

Under style transfer without using parallel data, Hu et al. (2017) control the sentiment and the tense of the generated text by learning a disentangled latent representation in a neural generative model. Ficler and Goldberg (2017) control several linguistic style aspects simultaneously by conditioning a recurrent neural network language model on specific style (professional, personal, length) and content (theme, sentiment) parameters. There has also been work on controlling style in neural machine translation. Sennrich et al. (2016) control the politeness of the translated text via side constraints, and the methods raised BLEU score by 3.2 points. Niu et al. (2017) control the level of formality of machine translation output by selecting phrases of a requisite formality level from the k-best list during decoding. They find that the best BLEU scores are obtained when the level of formality given as input to the machine translation system matches the nature of the text being translated. In the field of text simplification, more recently, Xu et al. (2016) learn large-scale paraphrase rules using bilingual texts whereas Kajiwara and Komachi (2016) build a monolingual parallel corpus using sentence similarity based on alignment between word embeddings.

In our work, we use a semi-supervised approach to generating text in a given style. During training our model, we append the source with a special token indicative of the style of the target sentence. These tokens are embedded into the source sentence representation and control target sequence generation via the attention mechanism. Sennrich et al. (2016) append <T> or <V> to the source text for distinguishing between the familiar (Latin Tu) and the polite (Latin Vos) second

person pronoun in the German output. Johnson et al. (2017) and Niu et al. (2018) concatenate parallel data of various language directions and mark the source with the desired output language to perform multilingual or bi-directional NMT. KOBUS et al. (2017) and Chu et al. (2017) add domain tags for domain adaptation in neural machine translation. Mima et al. (1997) improve rule-based machine translation by using extra-linguistic information such as speaker's role and gender. Lewis et al. (2015) and Niu and Carpuat (2016) equate style with domain, and train conversational machine translation systems by selecting in-domain (i.e. conversation-like) training data. Similarly, Wintner et al. (2017) and Michel and Neubig (2018) take an adaptation approach to personalize machine translation with gender-specific or speaker-specific data.

In summary, in this chapter we present a background study where we first define clarification questions and state their importance in human communication and therefore in human computer interactions. We discuss previous works in the general area of question generation and briefly explain previous major approaches to question generation. We also introduce the major neural network models that we use in our work. Finally, we discuss previous work related to text generation with stylistic variations.

Chapter 3: Dataset Creation

3.1 StackExchange Dataset

StackExchange is a network of online question answering websites about varied topics like Academia, Ubuntu operating system, Latex, etc. The sites are modeled after StackOverflow, a popular platform used for asking and answering questions on a wide range of topics in computer programming. On this platform, users frequently post issues they are facing with a particular topic and other users on the forum help resolve the issue. For instance, in Figure 3.1, a user posts an issue they are facing with installing an application on Ubuntu operating system. Another user comes and asks a clarification question in the comment section asking for the version of Ubuntu suggesting that that information is important for resolving the issue. The author subsequently comes back and edits the original post adding the version information.

The data dump of StackExchange contains timestamped information about the posts, comments on the post and the history of the revisions made to the post. We use this data dump to create our dataset of (*post, question, answer*) triples: where the *post* is the initial unedited post, the *question* is the comment containing a question and the *answer* is either the edit made to the post after the question or

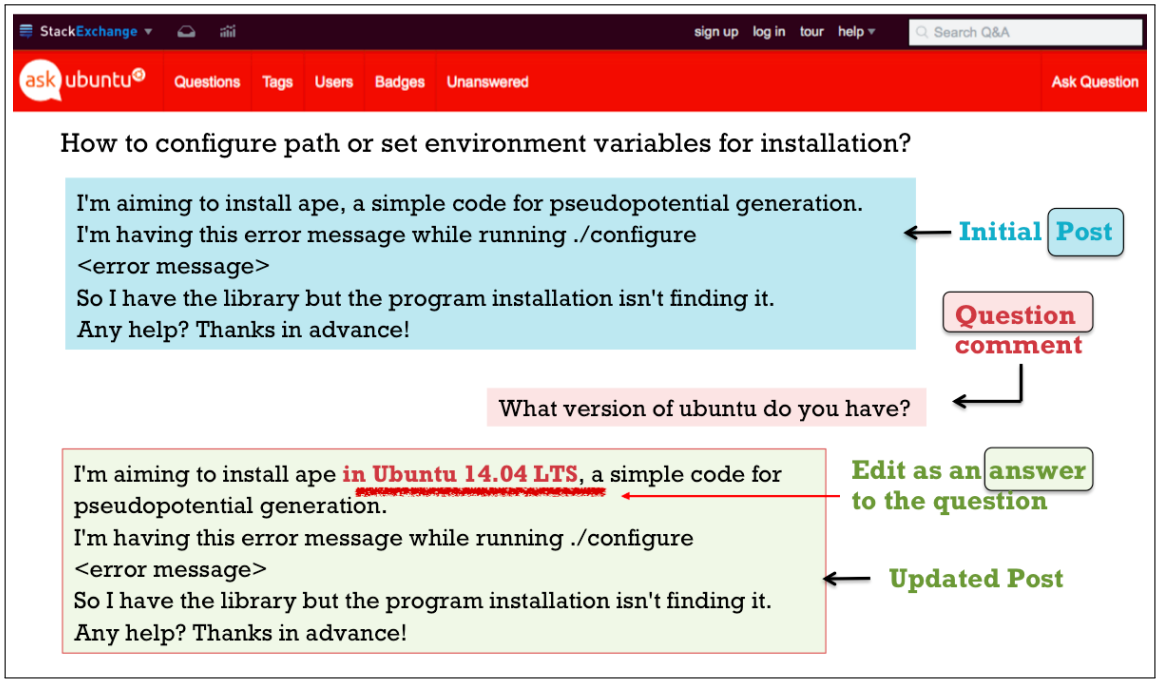

Figure 3.1: Example of a post on askubuntu.com where a user asked a clarification question in the comments section of the post following which the author of the post edited the post adding the missing information pointed out by the clarification question.

the author's response to the question in the comments section.[1]

Extract posts: We use the post histories to identify posts that have been updated by its author. We use the timestamp information to retrieve the initial unedited version of the post.

Extract questions: For each such initial version of the post, we use the timestamp information of its comments to identify the first comment made to the post. If the

[1]We use data from StackExchange; per license cc-by-sa 3.0, the data is "intended to be shared and remixed" (with attribution).

comment contains a question mark '?', we truncate the comment till the question mark '?' to retrieve the question part of the comment.

Filtering out questions: We find that about 7% of the questions are rhetoric that indirectly suggest a solution to the post. For e.g. *"have you considered installing X?"*. We do a manual analysis of these non-clarification questions and hand-crafted a few rules to remove them. We filter out questions that indirectly suggest a solution by ignoring questions that start with one of these phrases: "have you", "did you try", "can you try" or "could you try". We also ignore questions that contain one of the following words 'duplicate', 'upvote', 'downvote', 'vote', 'related', 'upvoted', 'downvoted' or 'edit'. We ignore questions that contain more than 20 tokens. Questions often start with "@username" when it is directed to a specific user. In these cases, we remove the initial part of the question corresponding to "@username".

Extract answers: We extract the answer to a clarification question in the following two ways:

1. *Edited post*: Authors tend to respond to a clarification question by editing their original post and adding the missing information. In order to account for edits made for other reasons like stylistic updates and grammatical corrections, we consider only those edits that are longer than four words. Authors can make multiple edits to a post in response to multiple clarification questions.To identify the edit made corresponding to the given question comment, we choose

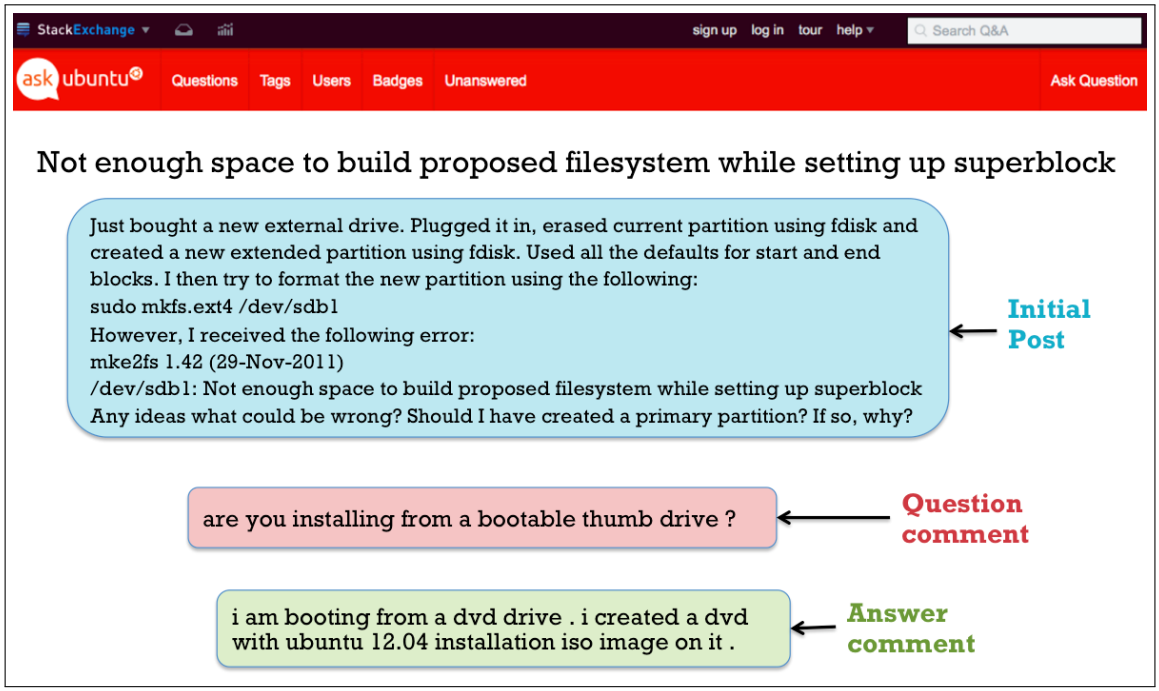

Figure 3.2: Example of a post on askubuntu.com where a user asked a clarification question in the comments section of the post and the author of the post answered the question as a subsequent comment.

the edit closest in time following the question.

2. *Response to the question*: Authors also respond to clarification questions as subsequent comments in the comment section (see Figure 3.2). We extract the first comment by the author following the clarification question as the answer to the question.

In cases where both the methods above yield an answer, we pick the one that is the most semantically similar to the question, where the measure of similarity is the cosine distance between the average word embeddings of the question and the answer.

	Train	Tune	Test
askubuntu	19,944	2493	2493
unix	10,882	1360	1360
superuser	30,852	3857	3856

Table 3.1: Table above shows the sizes of the train, tune and test split of our dataset for three domains.

We extract a total of 77,097 (*post, question, answer*) triples across three domains in StackExchange (Table 3.1). Although StackExchange consists of many sites, we choose the ones above because: a) the data dump available for them were moderately big in size to train a model on; b) these domains contain clarification questions that are generic enough to be useful for many different posts; and finally c) the three domains were close enough so that we could combine them and train on a larger dataset.

3.2 Analysis of StackExchange Dataset

How often are extracted questions clarifications? A natural question to our process of data creation would be how often is the extracted question a clarification question. We sample a set of 1000 questions from our dataset and design a crowdsourced task on Figure-Eight where given a question we ask annotators to choose whether the question was: (a) Asking for more information, (b) Providing an answer or a suggestion; or (c) Neither.[2] We collect three annotations per question. We find

[2]www.figureeight.com

that 91% of the questions were marked with option (a), 7% with option (b) and 2% with option (c). These numbers suggest that a large portion of the extracted questions are indeed "clarification questions". Additionally, we analyze the questions marked as "providing a solution" and find that majority of these started with one of the following phrases: *"have you", "did you try", "can you try", "could you try"*. We preprocess our dataset to remove all such instances.

How useful are clarifications questions? A clarification question is useful if it helps in generating an answer for a given post. Imagine a scenario in which a post goes unanswered for some time. Following this, a clarification question gets asked on this post and then the post gets an answer. Such a scenario will help showcase the usefulness of clarification questions. We estimate such a usefulness by calculating the following two probabilities for posts that have not received an answer within a week:

$$Pr(A|CQ) = \frac{\#(A|CQ)}{\#(A|CQ) + \#(\neg A|CQ)}$$
$$Pr(A|\neg CQ) = \frac{\#(A|\neg CQ)}{\#(A|\neg CQ) + \#(\neg A|\neg CQ)}$$

where:

$\#(A|CQ)$: # answered posts with a clarification question

$\#(\neg A|CQ)$: # unanswered posts with a clarification question

$\#(A|\neg CQ)$: # answered posts without a clarification question

$\#(\neg A|\neg CQ)$: # unanswered posts without a clarification question

Table 3.2 shows these probabilities for the three data domains. We can see that, overall, the likelihood of a post getting an answer with a clarification question is higher than the the likelihood of a post getting an answer without a clarification question.

Yes/No clarification questions We argue in the introduction of Chapter 4 that asking a question like "What version of Ubuntu do you have?" is more useful than asking a more specific question that might yield a Yes/No answer. This raises the question of how many clarification questions in our dataset are Yes/No questions. We manually inspect 100 randomly selected clarification questions in our dataset and find that 13 of them were Yes/No questions. This suggests that users, on these forums, tend to ask questions that are generic enough to elicit a useful answer more than a specific question.

Multiple clarification questions On analysis, we find that 35%-40% of the posts get asked multiple clarification questions. We include only the first clarification question to a post in our dataset since identifying if the following questions are clarifications or a part of a dialogue is non-trivial.

	askubuntu	unix	superuser
$Pr(A\|CQ)$	0.82	0.85	0.45
$Pr(A\|\neg CQ)$	0.77	0.80	0.34

Table 3.2: Likelihood of a post getting answered with and without a clarification question

3.3 Amazon Dataset

Amazon (amazon.com) is an online shopping platform where product sellers post descriptions of their products and users buy them online. Often when the given description is missing some important information, users ask questions in the frequently-asked-questions section of the product. For instance, Figure 3.3 shows the description of a cookware set under the "Home & Kitchen" category of amazon.com and a clarification question that asks if the cookware set is induction safe (i.e. works on induction stove).

McAuley and Yang (2016) introduced the Amazon question-answering dataset where each instance consists of a question asked about a product on amazon.com combined with other information (product ID, question type "Yes/No", answer type, answer and answer time). We extract the product ID, question and answer from this dataset. To obtain the description of the product, we use the Amazon reviews dataset (McAuley et al., 2015) which includes product ID and product description. We consider at most 10 questions for each product. This dataset includes several different product categories. We choose the `Home and Kitchen` category since it

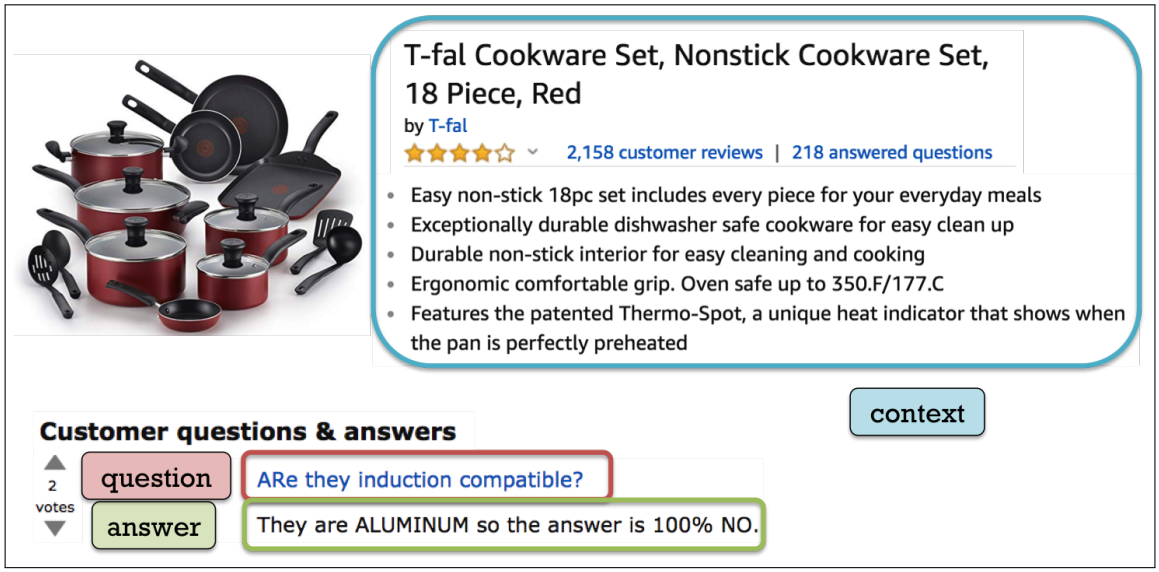

Figure 3.3: Example of a product description on amazon.com followed by a clarification question and an answer to the question.

contains a high number of questions. This dataset consists of 19,119 training, 2,435 validation and 2,305 test examples, and each product description contains between 3 and 10 questions (average: 7).

Chapter 4: Question Ranking Model

4.1 Introduction

In this chapter we describe our model for ranking clarification questions. A principal goal of asking questions is to fill information gaps, typically through clarification questions. We take the perspective that a good question is the one whose *likely answer* will be useful. Consider the exchange in Figure 4.1, in which an initial poster (who we call "Terry") asks for help configuring environment variables. This post is underspecified and a responder ("Parker") asks a clarifying question (a) below, but could alternatively have asked (b) or (c):

(a) What version of Ubuntu do you have?

(b) What is the make of your wifi card?

(c) Are you running Ubuntu 14.10 kernel 4.4.0-59-generic on an x86_64 architecture?

Parker should not ask (b) because an answer is unlikely to be useful; they should not ask (c) because it is too specific and an answer like "No" or "I do not know" gives little help. Parker's question (a) is much better: it is both likely to be useful, and is plausibly answerable by Terry.

In this work, we design a model to rank a candidate set of clarification questions by their usefulness to the given post. We imagine a use case (more discussion in §4.6)

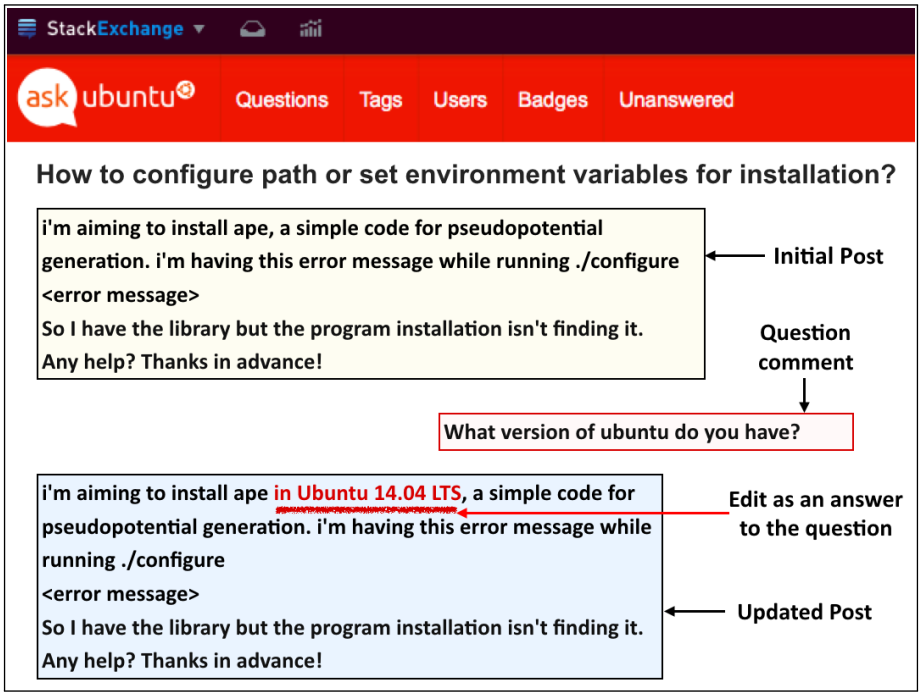

Figure 4.1: A post on an online Q & A forum "askubuntu.com" is updated to fill the missing information pointed out by the question comment.

in which, while Terry is writing their post, a system suggests a shortlist of questions asking for information that it thinks people like Parker might need to provide a solution, thus enabling Terry to immediately clarify their post, potentially leading to a much quicker resolution.

To develop our model we take inspiration from the decision theoretic framework of the Expected Value of Perfect Information (EVPI) (Avriel and Williams, 1970), a measure of the value of gathering additional information. In our setting, we use EVPI to calculate which question is most likely to elicit an answer that would make the post more informative (§ 4.2). Formally, for an input post p, we want to choose a question q that maximizes $\mathbb{E}_{a|p,q}[\mathbb{U}(p+a)]$, where a is a hypothetical answer and U is a function measuring the *utility* of post p if a were to be added to it. To

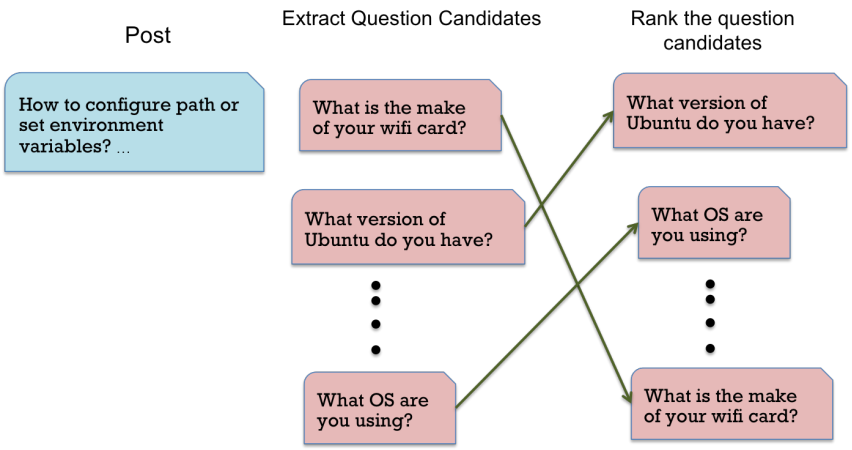

Figure 4.2: We formulate our ranking problem as, given a post, first extract a set of ten candidate questions and then rank them such that a more useful question would be higher up in the ranking

achieve this, we construct two models:

(1) an answer model, which estimates $\mathbb{P}[a \mid p, q]$, the likelihood of receiving answer a if one were to ask question q on post p (§4.2.2);

(2) an utility calculator, $\mathbb{U}(p)$, which measures the utility of the post (§4.2.3).

Given these two models, at prediction time we search over a shortlist of possible questions for that which maximizes the EVPI. We formulate this task as a ranking problem where given a post and a list of candidate questions, the task is to rank the questions such that a more useful question would be higher up in the ranking (refer Figure 4.2). The candidate list includes the "original" question asked to the post and nine other questions that we extract from posts that are similar to the given post.[1] Note that this setting is different from the distractor-based setting popularly

[1]Henceforth we refer to the question paired with the post as the "original" question

used in dialogue (Lowe et al., 2015) in that the nine other questions can include a good question.

We train our answer model and our utility calculator jointly based on (p, q, a) triples that we extract from StackExchange (§3.1), using its edit history (Figure 4.1). In the figure, the initial post fails to state what version of Ubuntu is being run. In response to Parker's question in the comments section, Terry, the author of the post, edits the post to answer Parker's clarification question. Terry might also choose to answer the clarification question in the subsequent comment. We extract the initial post p, question posted in the comments section q, and edit to the original post or the comment following the clarification question comment as answer a to form our (p, q, a) triples.

We evaluate our models using human judgments that we collect on Upwork.[2] We ask annotators to select what they thought was the *single* best question to ask, and additionally mark as "valid" any other questions that they thought would be okay to ask (§ 4.3). We evaluate models both on the task of returning the original clarification question and also on the task of picking any of the candidate clarification questions marked as good by experts. We find that our EVPI model outperforms the baseline models when evaluated against expert human annotations. We include a few examples of human annotations along with our model performance on them in § 4.5. We have released our dataset of ∼77K (p, q, a) triples and the expert annotations on 500 triples to help facilitate further research in this task.[3]

[2]https://www.upwork.com

[3]`https://github.com/raosudha89/ranking_clarification_questions`

4.2 Model description

We build a neural network model inspired by the theory of expected value of perfect information (EVPI). EVPI is a measurement of: if I were to acquire information X, how useful would that be to me? However, because we haven't acquired X yet, we have to take this quantity in expectation over all possible X, weighted by each X's likelihood. In our setting, for any given question q_i that we can ask, there is a set A of possible answers that could be given. For each possible answer $a_j \in A$, there is some probability of getting that answer, and some utility if that were the answer we got. The value of this question q_i is the expected utility, over all possible answers:

$$EVPI(q_i|p) = \sum_{a_j \in A} \mathbb{P}[a_j|p, q_i]\mathbb{U}(p + a_j) \tag{4.1}$$

In Eq 4.1, p is the post, q_i is a potential question from a set of candidate questions Q and a_j is a potential answer from a set of candidate answers A. Here, $\mathbb{P}[a_j|p, q_i]$ measures the probability of getting an answer a_j given an initial post p and a clarifying question q_i, and $\mathbb{U}(p + a_j)$ is a utility function that measures how much more complete p would be if it were augmented with answer a_j. The modeling question then is how to model:

1. The probability distribution $\mathbb{P}[a_j|p, q_i]$ and
2. The utility function $\mathbb{U}(p + a_j)$.

In our work, we represent both using neural networks over the appropriate inputs. We train the parameters of the two models jointly to minimize a joint loss defined

such that an answer that has a higher potential of increasing the utility of a post gets a higher probability.

Figure 5.1 describes the behavior of our model during test time. Given a post p, we generate a set of candidate questions and a set of candidate answers (§ 4.2.1). Given a post p and a question candidate q_i, we calculate how likely is this question to be answered using one of our answer candidates a_j (§ 4.2.2). Given a post p and an answer candidate a_j, we calculate the utility of the updated post i.e. $\mathbb{U}(p + a_j)$ (§ 4.2.3). We compose these modules into a joint neural network that we optimize end-to-end over our data (§ 4.2.4).

4.2.1 Question & answer candidate generator

Given a post p, our first step is to generate a set of question and answer candidates. One way that humans learn to ask questions is by looking at how others ask questions in a similar situation. Using this intuition we generate question candidates for a given post by identifying posts similar to the given post and then looking at the questions asked to those posts. For identifying similar posts, we use Lucene, a software extensively used in information retrieval for extracting documents relevant to a given query from a pool of documents.[4] Lucene implements a variant of the term frequency-inverse document frequency (TF-IDF) model to score the extracted documents according to their relevance to the query. We use Lucene to find the top 10 posts most similar to a given post from our dataset (§ 3.1). We consider the questions asked to these 10 posts as our set of question candidates Q

[4] `https://lucene.apache.org/`

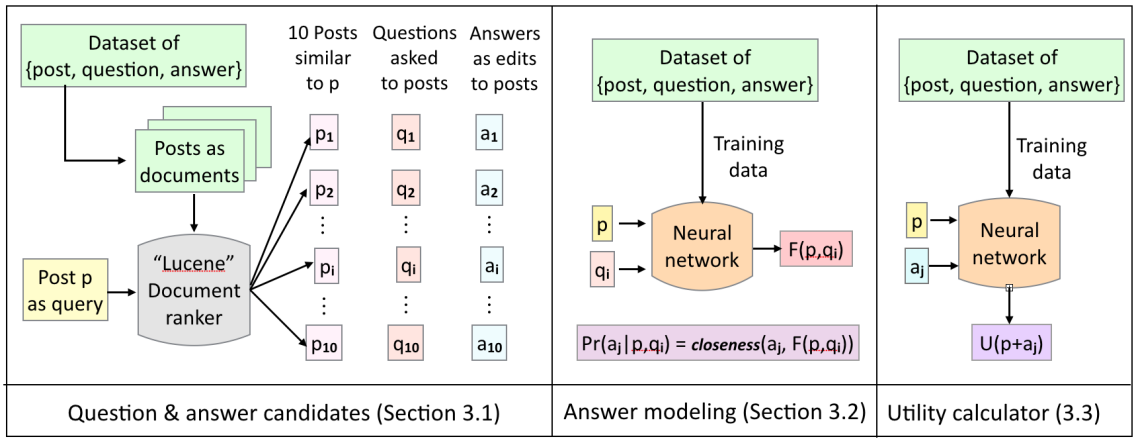

Figure 4.3: The behavior of our model during test time: Given a post p, we retrieve 10 posts similar to post p using Lucene. The questions asked to those 10 posts are our question candidates Q and the edits made to the posts in response to the questions (or the author's response to the question in the comments section) are our answer candidates A. For each question candidate q_i, we generate an answer representation $F(p, q_i)$ and calculate how close is the answer candidate a_j to our answer representation $F(p, q_i)$. We then calculate the utility of the post p if it were updated with the answer a_j. Finally, we rank the candidate questions Q by their expected utility given the post p (Eq 4.1).

and the edits made to the posts in response to the questions as our set of answer candidates A. Since the top-most similar candidate extracted by Lucene is always the original post itself, the original question and answer paired with the post is always one of the candidates in Q and A. § 3.1 describes in detail the process of extracting the (*post, question, answer*) triples from the StackExchange datadump.

4.2.2 Answer modeling

Given a post p and a question candidate q_i, our second step is to calculate how likely is this question to be answered using one of our answer candidates a_j. We first generate an answer representation by combining the neural representations of the post and the question using a function $F_{ans}(\bar{p}, \bar{q}_i)$ (details in § 4.2.4). Given such a representation, we measure the distance between this answer representation and one of the answer candidates a_j using the function below:

$$dist(F_{ans}(\bar{p},\ \bar{q}_i),\ \hat{a}_j) = 1 - cos\_sim(F_{ans}(\bar{p},\ \bar{q}_i),\ \hat{a}_j)$$

The likelihood of an answer candidate a_j being the answer to a question q_i on post p is finally calculated by combining this distance with the cosine similarity between the question q_i and the question q_j paired with the answer candidate a_j:

$$\mathbb{P}[a_j|p, q_i] = \exp^{-dist(F_{ans}(\bar{p},\ \bar{q}_i),\ \hat{a}_j)} * cos\_sim(\hat{q}_i,\ \hat{q}_j) \tag{4.2}$$

where $\hat{a}_j$, $\hat{q}_i$ and $\hat{q}_j$ are the average word vector of a_j, q_i and q_j respectively (details in §4.2.4) and cos_sim is the cosine similarity between the two input vectors.

We model our answer generator using the following intuition: a question can be asked in several different ways. For e.g. in Figure 4.1, the question "What version of Ubuntu do you have?" can be asked in other ways like "What version of operating system are you using?", "Version of OS?", etc. Additionally, for a given post and a question, there can be several different answers to that question. For instance, "Ubuntu 14.04

LTS", "Ubuntu 12.0", "Ubuntu 9.0", are all valid answers. To generate an answer representation capturing these generalizations, we train our answer generator on our triples dataset (§ 3.1) using the loss function below:

$$\text{loss}_{\text{ans}}(p_i, q_i, a_i, Q_i) = dist(F_{ans}(\bar{p}_i, \bar{q}_i),\ \hat{a}_i) + \sum_{j \in Q} \Big(dist(F_{ans}(\bar{p}_i, \bar{q}_i),\ \hat{a}_j) * cos\_sim(\hat{q}_i,\ \hat{q}_j) \Big) \tag{4.3}$$

where, $\hat{a}$ and $\hat{q}$ is the average word vectors of a and q respectively (details in § 4.2.4), cos_sim is the cosine similarity between the two input vectors.

This loss function can be explained using the example in Figure 4.4. Question q_i is the question paired with the given post p_i. In Eq 4.3, the first term forces the function $F_{ans}(\bar{p}_i, \bar{q}_i)$ to generate an answer representation as close as possible to the correct answer a_i. Now, a question can be asked in several different ways. Let Q_i be the set of candidate questions for post p_i, retrieved from the dataset using Lucene (§ 4.2.1). Suppose a question candidate q_j is very similar to the correct question q_i (i.e. $cos\_sim(\hat{q}_i,\ \hat{q}_j)$ is near zero). Then the second term forces the answer representation $F_{ans}(\bar{p}_i, \bar{q}_i)$ to be close to the answer a_j corresponding to the question q_j as well. Thus in Figure 4.4, the answer representation will be close to a_j (since q_j is similar to q_i), but may not be necessarily close to a_k (since q_k is dissimilar to q_i). This is similar to the idea of co-occurrence smoothing (Essen and Steinbiss, 1992; Resnik, 1993), a method which combines prediction information of distinct words based on their distributional similarity in order to smooth language models.

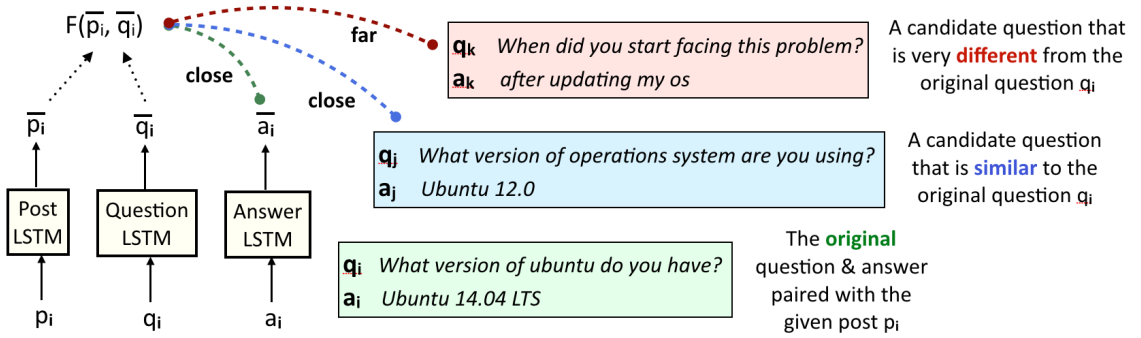

Figure 4.4: Training of our answer generator. Given a post p_i and its question q_i, we generate an answer representation that is not only close to its original answer a_i, but also close to one of its candidate answers a_j if the candidate question q_j is close to the original question q_i.

4.2.3 Utility calculator

Given a post p and an answer candidate a_j, the third step is to calculate the utility of the updated post i.e. $\mathbb{U}(p{+}a_j)$. As expressed in Eq 4.1, this utility function measures how useful it would be if a given post p were augmented with an answer a_j paired with a different question q_j in the candidate set. Although theoretically, the utility of the updated post can be calculated only using the given post (p) and the candidate answer (a_j), empirically we find that our neural EVPI model performs better when the candidate question (q_j) paired with the candidate answer is a part of the utility function. We attribute this to the fact that much information about whether an answer increases the utility of a post is also contained in the question asked to the post. We train our utility calculator using our dataset of (p, q, a) triples (§3.1). We label all the (p_i, q_i, a_i) pairs from our triples dataset with label $y = 1$. To

get negative samples, we make use of the answer candidates generated using Lucene as described in § 4.2.1. For each $a_j \in A_i$, where A_i is the set of answer candidates for post p_i, we label the pair (p_i, q_j, a_j) with label $y = 0$, except for when $a_j = a_i$. Thus, for each post p_i in our triples dataset, we have one positive sample and nine negative samples. This idea of using implicit negative evidence for training is similar to the notion of contrastive estimation (Smith and Eisner, 2005). It should be noted that this is a noisy labelling scheme since a question not paired with the original question in our dataset can often times be a *good* question to ask to the post (§ 4.3). However, since we do not have annotations for such other good questions at train time, we assume such a labelling.

Given a post p_i and an answer a_j paired with the question q_j, we combine their neural representations using a function $F_{util}(\bar{p_i}, \bar{q_j}, \bar{a_j})$ (details in § 4.2.4). The utility of the updated post is then defined as $\mathbb{U}(p_i + a_j) = \sigma(F_{util}(\bar{p_i}, \bar{q_j}, \bar{a_j}))$, where σ is the sigmoid function. We want this utility to be close to 1 for all the positively labelled (p, q, a) triples and close to 0 for all the negatively labelled (p, q, a) triples. We therefore define our loss using the binary cross-entropy formulation below:

$$\text{loss}_{\text{util}}(y_i, \bar{p_i}, \bar{q_j}, \bar{a_j}) = y_i \log(\sigma(F_{util}(\bar{p_i}, \bar{q_j}, \bar{a_j}))) \tag{4.4}$$

4.2.4 Our joint neural network model

Our fundamental representation is based on recurrent neural networks over word embeddings. We obtain the word embeddings using the GloVe (Pennington et al., 2014) model trained on the entire datadump of StackExchange. In Eq 4.2 and

Eq 4.3, the average word vector representations $\hat{q}$ and $\hat{a}$ are obtained by averaging the GloVe word embeddings for all words in the question and the answer respectively. Given an initial post p, we generate a post neural representation $\bar{p}$ using a post LSTM (long short-term memory architecture) (Hochreiter and Schmidhuber, 1997). The input layer consists of word embeddings of the words in the post which is fed into a single hidden layer. The output of each of the hidden states is averaged together to get our neural representation $\bar{p}$. Similarly, given a question q and an answer a, we generate the neural representations $\bar{q}$ and $\bar{a}$ using a question LSTM and an answer LSTM respectively. We define the function F_{ans} in our answer model as a feedforward neural network with five hidden layers on the inputs $\bar{p}$ and $\bar{q}$ as shown in Figure 4.5. Likewise, we define the function F_{util} in our utility calculator as a feedforward neural network with five hidden layers on the inputs $\bar{p}$, $\bar{q}$ and $\bar{a}$. We train the parameters of the three LSTMs corresponding to p, q and a, and the parameters of the two feedforward neural networks jointly to minimize the sum of the loss of our answer model (Eq 4.3) and our utility calculator (Eq 4.4) over our entire dataset:

$$\sum_i \sum_j \text{loss}_{\text{ans}}(\bar{p}_i, \bar{q}_i, \bar{a}_i, Q_i) + \text{loss}_{\text{util}}(y_i, \bar{p}_i, \bar{q}_j, \bar{a}_j) \tag{4.5}$$

Given such an estimate $\mathbb{P}[a_j|p, q_i]$ of an answer and a utility $\mathbb{U}(p + a_j)$ of the updated post, we rank the candidate questions by their value as calculated using Eq 4.1. The remaining question, then, is how to get data that enables us to train our answer model and our utility calculator. Given data, the training becomes a multitask learning problem, where we learn simultaneously to predict utility and to

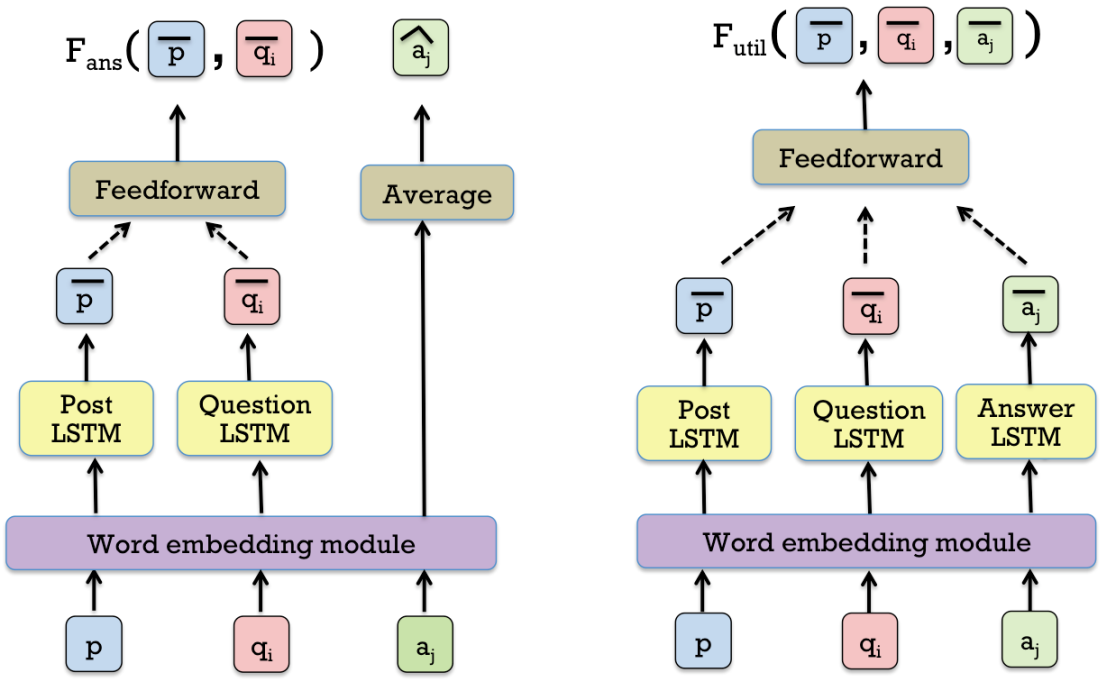

Figure 4.5: Left: F_{ans} computed using a feedfoward neural network over post LSTM $\bar{p}$ and question LSTM $\bar{q}$ representations and $\hat{a}$ computed using average word embeddings over words in the answer. Right: F_{util} computed using a feedforward neural network over post LSTM $\bar{p}$, question LSTM $\bar{q}$ and answer LSTM $\bar{a}$ representations.

estimate the probability of answers.

4.3 Evaluation design

We define our task as given a post p, and a set of candidate clarification questions Q, rank the questions according to their usefulness to the post. Since the candidate set includes the original question q that was asked to the post p, one possible approach to evaluation would be to look at how often the original question is ranked higher up in the ranking predicted by a model. However, there are two

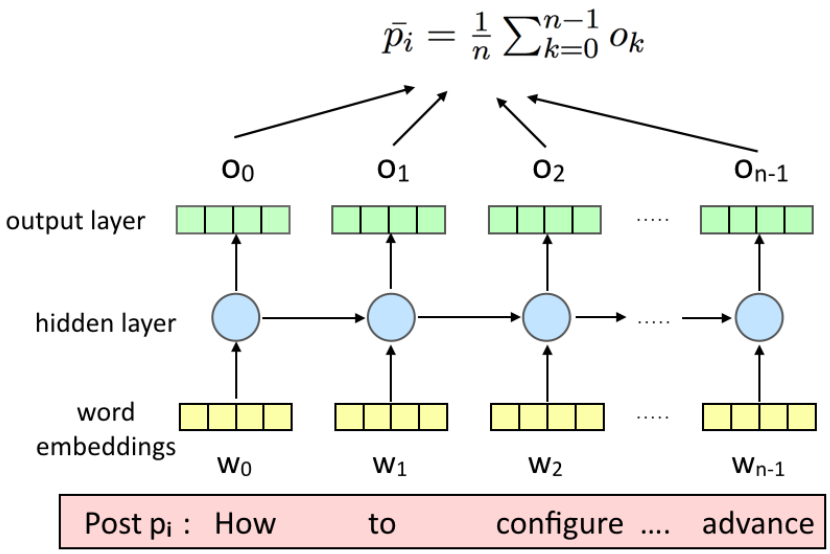

Figure 4.6: Our LSTM architecture on a post p_i. The input layer consists of pre-trained word embeddings of the words in the post which is fed into a single hidden layer. The output o_k of each of the hidden states is averaged together to get our neural representation $\bar{p_i}$

problems to this approach: 1) Our dataset creation process is noisy. The original question paired with the post may not be a useful question. For e.g. *"are you seriously asking this question?"*, *"do you mind making that an answer?"*.[5] 2) The nine other questions in the candidate set are obtained by looking at questions asked to posts that are similar to the given post.[6] This greatly increases the possibility of some other question(s) being more useful than the original question paired with the post. This motivates an evaluation design that does not rely solely on the original question but also uses human judgments. We randomly choose a total of

[5]Data analysis in Chapter 3 suggests 9% of the questions are not useful.

[6]Note that this setting is different from the distractor-based setting popularly used in dialogue (Lowe et al., 2015) where the distractor candidates are chosen randomly from the corpus.

500 examples from the test sets of the three domains proportional to their train set sizes (`askubuntu:160`, `unix:90` and `superuser:250`) to construct our evaluation set.

4.3.1 Annotation scheme

Due to the technical nature of the posts in our dataset, identifying useful questions requires technical experts. We recruit 10 such experts on Upwork who have prior experience in Unix based operating system administration. As a training process, we first ask the annotators to annotate a sample of 5 examples and provide them with feedback and additional guidance. We also ask annotators to rate their confidence in {1: Educated guess, 2: Pretty sure, 3: Quite sure}. The confidence on 17% of the annotations was rated as low, 47% was rated as medium and 37% was rated as high.

We provide the annotators with a post and a randomized list of the ten question candidates obtained using Lucene (§ 4.2.1) and ask them to select a *single* "best" (B) question to ask, and additionally mark as "valid" (V) other questions that they thought would be okay to ask in the context of the original post. We enforce that the "best" question be always marked as a "valid" question. We group the 10 annotators into 5 pairs and assign the same 100 examples to the two annotators in a pair.

4.3.2 Annotation analysis

We calculate the inter-annotator agreement on the "best" and the "valid" annotations using Cohen's Kappa measurement. When calculating the agreement on the "best" in the strict sense, we get a low agreement of 0.15. However, when we relax this to a case where the question marked as"best" by one annotator is marked as "valid" by another, we get an agreement of 0.87. The agreement on the "valid" annotations, on the other hand, was higher: 0.58. We calculate this agreement on the binary judgment of whether a question was marked as valid by the annotator.

Given these annotations, we calculate how often is the original question marked as "best" or "valid" by the two annotators. We find that 72% of the time one of the annotators mark the original as the "best", whereas only 20% of the time both annotators mark it as the "best" suggesting against an evaluation solely based on the original question. On the other hand, 88% of the time one of the two annotators mark it as a "valid" question confirming the noise in our training data.[7]

Figure 4.7 shows the distribution of the counts of questions in the intersection of "valid" annotations (blue legend). We see that about 85% of the posts have more than 2 valid questions and 50% have more than 3 valid questions. The figure also shows the distribution of the counts when the original question is removed from the intersection (red legend). Even in this set, we find that about 60% of the posts have more than two valid questions. These numbers suggests that the candidate set of questions retrieved using Lucene (§ 4.2.1) very often contains useful clarification

[7] 76% of the time both the annotators mark it as a "valid".

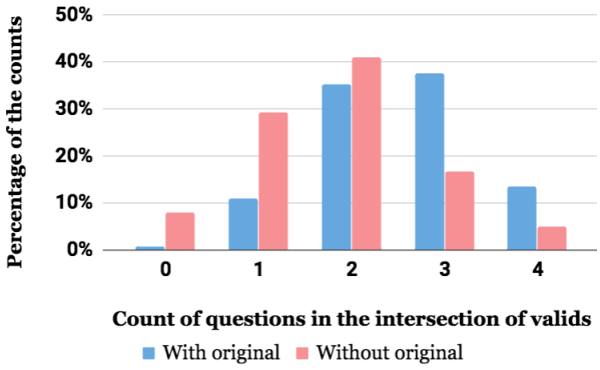

Figure 4.7: Distribution of the count of questions in the intersection of the "valid" annotations.

questions.

4.4 Experimental results

Our primary research questions that we evaluate experimentally are:

1. Does a neural architecture with learned representations improve upon a simple bag-of-ngrams baseline?
2. Does the expected value of perfect information (EVPI) formalism provide leverage over a similarly expressive feedforward network?
3. Are answers useful in identifying the right question?
4. How do the models perform when evaluated on the candidate questions excluding the original?

4.4.1 Baseline methods

We compare our model with following baselines:

	$B1 \cup B2$				$V1 \cap V2$				Original
Model	p@1	p@3	p@5	MAP	p@1	p@3	p@5	MAP	p@1
Random	17.5	17.5	17.5	35.2	26.4	26.4	26.4	42.1	10.0
Bag-of-ngrams	19.4	19.4	18.7	34.4	25.6	27.6	27.5	42.7	10.7
Community QA	23.1	21.2	20.0	40.2	33.6	30.8	29.1	47.0	18.5
Neural (p, q)	21.9	20.9	19.5	39.2	31.6	30.0	28.9	45.5	15.4
Neural (p, a)	24.1	**23.5**	20.6	41.4	32.3	**31.5**	29.0	46.5	18.8
Neural (p, q, a)	25.2	**22.7**	**21.3**	42.5	34.4	**31.8**	**30.1**	47.7	20.5
EVPI	**27.7**	**23.4**	**21.5**	**43.6**	**36.1**	**32.2**	**30.5**	**49.2**	**21.4**

Table 4.1: Model performances on 500 samples when evaluated against the union of the "best" annotations ($B1 \cup B2$), intersection of the "valid" annotations ($V1 \cap V2$) and the original question paired with the post in the dataset. The difference between the bold and the non-bold numbers is statistically significant with $p < 0.05$ as calculated using bootstrap test. p@k is the precision of the k questions ranked highest by the model and MAP is the mean average precision of the ranking predicted by the model.

Random: Given a post, we randomly permute its set of 10 candidate questions uniformly. We take the average over 1000 random permutations.

Bag-of-ngrams: Given a post and a set of 10 question and answer candidates, we construct a bag-of-ngrams representation for the post, question and answer. We train the baseline on all the positive and negative candidate triples (same as in our utility calculator (§ 4.2.3)) to minimize hinge loss on misclassification error using cross-product features between each of $(p,\ q)$, $(q,\ a)$ and $(p,\ a)$. We tune the ngram length and choose n=3 which performs best on the tune set. The question candidates are finally ranked according to their predictions for the positive label.

Community QA: The recent SemEval2017 Community Question-Answering (CQA) (Nakov et al., 2017) included a subtask for ranking a set of comments according to their relevance to a given post in the Qatar Living forum. [8] Nandi et al. (2017), winners of this subtask, developed a logistic regression model using features based on string similarity, word embeddings, etc. We train this model on all the positively and negatively labelled (p, q) pairs in our dataset (same as in our utility calculator (§ 4.2.3), but without a). We use a subset of their features relevant to our task. Details in § 4.4.2.

Neural baselines: We construct the following neural baselines based on the LSTM representation of their inputs (as described in § 4.2.4):

1. **Neural(p, q):** Input is concatenation of $\bar{p}$ and $\bar{q}$.

[8] http://www.qatarliving.com/forum

2. **Neural(p, a):** Input is concatenation of $\bar{p}$ and $\bar{a}$.

3. **Neural(p, q, a):** Input is concatenation of $\bar{p}$, $\bar{q}$ and $\bar{a}$.

Given these inputs, we construct a fully connected feedforward neural network with 10 hidden layers and train it to minimize the binary cross entropy across all positive and negative candidate triples (same as in our utility calculator (§ 4.2.3)). We use 10 (double the number of hidden layers used in our EVPI model) hidden layers to ensure that the improvement in our EVPI model is not merely because of the increased number of parameters in the EVPI model. The major difference between the neural baselines and our EVPI model is in the loss function: the EVPI model is trained to minimize the joint loss between the answer model (defined on $F_{ans}(p, q)$ in Eq 4.3) and the utility calculator (defined on $F_{util}(p, q, a)$ in Eq 4.4) whereas the neural baselines are trained to minimize the loss directly on $F(p, q)$, $F(p, a)$ or $F(p, q, a)$.

4.4.2 Implementation details

Preprocessing: We tokenize the raw text in our post, question and answer using the NLTK tokenizer. We restrict the post to its first 300 tokens and the question and answer to first 40 tokens. In our work, we choose these token lengths based on the average lengths of posts and questions in the dataset. However, it is an open research question as to how would changing these token lengths influence the model predictions.

Word embedding model: Each post, question and answer in our dataset is represented using embeddings. To generate these embeddings, we train 200 dimensional word embeddings using GloVe on the 3 billion token datadump of StackExchange. Since the total number of tokens in the datadump is large, we use an unusually large threshold frequency of 100 to create a vocabulary of 250,000 tokens. All tokens with a frequency of less than 100 in our dataset get assigned an 'UNK' token.

Model hyperparameters: The hidden layers in all the neural models are of size 200. We use ReLU non-linearity as our activation function between the hidden layers. We use a batch size of 128. We train the models for up to 14 epochs and at test time we use the predictions of the epoch where the performance on the tune set is the best.

Community QA baseline: We use the implementation provided by the winning team of the SemEval2017 Community Question-Answering (cQA) subtask 3.[9] Their original model contains six feature groups: string similarity features, word embedding features, topic modeling features, keyword features, meta data features and dialogue identification features. Since we do not have information about the latter three features in our dataset, we use only the first three features and train a logistic regression model to obtain the confidence scores on the positive labels.

[9]https://github.com/TitasNandi/cQARank

4.4.3 Results

4.4.3.1 Evaluating against expert annotations

We first describe the results of the different models when evaluated against the expert annotations we collect on 500 samples (§4.3). Since the annotators had a low agreement on a single best, we evaluate against the union of the "best" annotations ($B1 \cup B2$ in Table 4.1) and against the intersection of the "valid" annotations ($V1 \cap V2$ in Table 4.1).

Among non-neural baselines, we find that the bag-of-ngrams baseline performs slightly better than random but worse than all the other models. The Community QA baseline, on the other hand, performs better than the neural baseline (Neural (p, q)), both of which are trained without using the answers. The neural baselines with answers (Neural(p, q, a) and Neural(p, a)) outperform the neural baseline without answers (Neural(p, q)), showing that answer helps in selecting the right question.

More importantly, EVPI outperforms the Neural (p, q, a) baseline across most metrics. Both models use the same information regarding the true question and answer and are trained using the same number of model parameters.[10] However, the EVPI model, unlike the neural baseline, additionally makes use of alternate question and answer candidates to compute its loss function. This shows that when the candidate set consists of questions similar to the original question, summing over their utilities gives us a boost.

[10]We use 10 hidden layers in the feedforward network of the neural baseline and five hidden layers each in the two feedforward networks F_{ans} and F_{util} of the EVPI model.

We can interpret the absolute numbers obtained by our best (EVPI) model in a real world setting as follows: Given 10 candidate questions obtained from Lucene, around 28% of the time, the top ranked question is the best question whereas around 36% of the time, the top ranked question is a valid question. Likewise, around 23% of the time, the top three questions are the best questions whereas around 32% of the time, the top three questions are valid questions. Although these absolute numbers are relatively low, in this work, we set the baseline for this novel task and hope that this work will encourage future work in this space.

4.4.3.2 Evaluating against the original question

The last column in Table 4.1 shows the results when evaluated against the original question paired with the post. The bag-of-ngrams baseline performs similar to random, unlike when evaluated against human judgments. The Community QA baseline again outperforms Neural(p, q) model and comes very close to the Neural (p, a) model.

As before, the neural baselines that make use of the answer outperform the one that does not use the answer and the EVPI model performs significantly better than Neural(p, q, a).

4.4.3.3 Excluding the original question

In the preceding analysis, we considered a setting in which the "ground truth" original question was in the candidate set Q. While this is a common evaluation

Model	$B1 \cup B2$				$V1 \cap V2$			
	p@1	p@3	p@5	MAP	p@1	p@3	p@5	MAP
Random	17.4	17.5	17.5	26.7	26.3	26.4	26.4	37.0
Bag-of-ngrams	16.3	18.9	17.5	25.2	26.7	28.3	26.8	37.3
Community QA	22.6	20.6	18.6	29.3	30.2	29.4	27.4	38.5
Neural (p,q)	20.6	20.1	18.7	27.8	29.0	29.0	27.8	38.9
Neural (p,a)	22.6	20.1	18.3	28.9	30.5	28.6	26.3	37.9
Neural (p,q,a)	22.2	21.1	19.9	28.5	29.7	29.7	28.0	38.7
EVPI	23.7	21.2	19.4	29.1	31.0	30.0	28.4	39.6

Table 4.2: Model performances on 500 samples when evaluated against the union of the "best" annotations ($B1 \cup B2$) and intersection of the "valid" annotations ($V1 \cap V2$), with the original question excluded. The difference between all numbers except the random and bag-of-ngrams are statistically insignificant.

framework in dialog response selection (Lowe et al., 2015), it is overly optimistic. We, therefore, evaluate against the "best" and the "valid" annotations on the nine other question candidates. We find that the neural models beat the non-neural baselines. However, the differences between all the neural models are statistically insignificant. Results are shown in Table 4.2

4.5 Example outputs

To understand the behavior of our EVPI model, we have included three example outputs in Table 4.4 one each from the three domains in our dataset. The first example is a case where the EVPI model predicts both the "best" and the "valid" questions higher in its ranking. The original poster is facing some issue they call the "suspend resume" issue. The post is unclear on what problem the poster is facing. Hence the "best" question asks for that information. In the second example, the model predicts one of the "valid" questions higher up in its ranking but fails to predict the "best" question. The model predicts "why would you need this" with very high probability likely because it is a very generic question, unlike the question marked as "best" by the annotator which is too specific. In the third example, the model again predicts a very generic question which is also marked as "valid" by the annotator. These examples suggest that the model is good at correctly predicting generic questions, but not at predicting very specific questions.

4.6 Conclusion

In this chapter we describe a novel model for the task of ranking clarification questions. Our model integrates well-known deep network architectures with the classic notion of expected value of perfect information, which effectively models a pragmatic choice on the part of the questioner: how do I *imagine* the other party would answer if I were to ask this question. Such pragmatic principles have

recently been shown to be useful in other tasks as well (Andreas and Klein, 2016; Golland et al., 2010; Orita et al., 2015; Smith et al., 2013). One can naturally extend our EVPI approach to a full reinforcement learning approach to handle multi-turn conversations.

Our results show that the EVPI model is a promising formalism for the question generation task. In order to move to a full system that can help users like Terry write better posts, the model needs to be able to generalize. For instance, if our model has access to posts in the training data that only discuss Ubuntu operating system, then our ranking model will never be able to generate a question such as "What version of Windows are you using?" even if it has seen questions such as "What version of Ubuntu are you using?". Another issue with our ranking model is that it relies on Lucene to retrieve a good initial set of candidate questions. In order to be able to exploit the usefulness of our model to the fullest, we therefore move from question ranking to a question generation task setup where given a context, we develop a model to generate a question from scratch. In our next chapter, we describe our question generation model that is based on sequence-to-sequence neural network models that have recently proven to be effective for several language generation tasks (Serban et al., 2016b; Sutskever et al., 2014; Yin et al., 2016).

Title:	**Ubuntu 15.10 instant resume from suspend**
Post:	I have an ASUS desktop PC that I decided to install Ubuntu onto. I have used Linux before, specifically for 3 years in High School. I have never encountered suspend resume issues on Linux before. It appears that my PC is resuming from suspend on Ubuntu 15.10 I am not sure what is causing this, but my hardware is as follows: Intel Core i5 4460 @ 3.2 GHz 2 TB Toshiba 7200 RPM disk 8 GB DDR3 RAM Corsair CX 500 Power Supply AMD Radeon R9 270X Graphics - 4 Gigs ASUS Motherboard for OEM builds VIA technologies USB 3.0 Hub Realtek Network Adapter Any help is greatly appreciated. I haven't worked with Linux in over a year, and as I plan to pursue a career in Comp Science (specifically through internshipsl) and this is a problem, as I don't want to drive the power bill up. (Even though I don't pay it, my parents do.)
✓0.87	does suspend - resume work as expected ?
✓0.71	what , specifically , is the problem you want help with ?
✓0.70	the suspend problem exits only if a virtual machines is running ?
0.67	is the pasted workaround still working for you ?
0.57	just wondering if you got a solution for this ?
0.50	we *could* try a workaround , with a keyboard shortcut . would that interest you ?
0.49	did you restart the systemd daemon after the changes 'sudo restart systemd-logind' ?
0.49	does running 'sudo modprobe -r psmouse ; sleep 1 ; sudo modprobe psmouse' enable the touchpad ?
0.49	2 to 5 minutes ?
0.49	does it work from the menu or not ?

Table 4.3: Example of human annotation from the askubuntu domain of our dataset. The questions are sorted by expected utility, given in the first column. The "best" annotation is marked with black ticks ✓and the "valid"' annotations are marked with grey ticks ✓.

Title:	**Frozen Linux Recovery Without SysReq**
Post:	RHEL system has run out of memory and is now frozen. The SysReq commands are not working, so I am not even sure that /proc/sys/kernel/sysrq is set to 1. Is there any other "safe" way I can reboot w/out power cycling?
0.91	why would you need this ?
✓0.77	maybe you need to use your 'fn' key when pressing print screen ?
0.59	do you have sudo rights on this computer ?
0.55	are you sure sysrq is enabled on your machine ?
0.52	did you look carefully at the logs when you rebooted after it hung ?
0.51	i assume you have data open which needs to be saved ?
✓0.50	define " frozen " . did it panic ? or did something else happen ?
✓0.50	maybe you need to use your 'fn' key when pressing print screen ?
0.50	tried ctrl + alt + f2 ?
0.49	does the script process 1 iteration successfully ?
0.49	laptop or desktop ?
Title:	**How to flash a USB drive?.**
Post:	I have a 8 GB Sandisk USB drive. Recently it became write somehow. So I searched in Google and I tried to remove the write protection through almost all the methods I found. Unfortunately nothing worked. So I decided to try some other ways. Some said that flashing the USB drive will solve the problem. But I don't know how. So how can it be done ?
✓1.01	what file system was the drive using ?
1.00	was it 16gb before or it has been 16mb from the first day you used it ?
✓0.74	which os are you using ? which file system is used by your pen drive ?
0.64	what operation system you use ?
0.51	can you narrow 'a hp usb down ' ?
0.50	could the device be simply broken ?
0.50	does it work properly on any other pc ?
0.50	usb is an interface , not a storage device . was it a flash drive or a portable disk ?
0.49	does usb flash drive tester have anything useful to say about the drive ?
✓0.49	your drive became writeable ? or read-only ?

Table 4.4: Examples of human annotation from the unix and superuser domain of our dataset. The questions are sorted by expected utility, given in the first column. The "best" annotation is marked with black ticks ✓and the "valid"' annotations are marked with grey ticks ✓.

Chapter 5: Question Generation Model

5.1 Introduction

In this chapter, we describe our clarification question generation model which given a context, generates a question one word at a time. Our clarification question generation model builds on the sequence-to-sequence approach that has proven effective for several language generation tasks (Du et al., 2017; Serban et al., 2016b; Sutskever et al., 2014; Yin et al., 2016). Unfortunately, training a sequence-to-sequence model directly on (context, question) pairs yields questions that are highly generic[1], corroborating a common finding in dialog systems (Li et al., 2016b). Our goal is to be able to generate clarification questions that are useful *and* specific.

To achieve this, we begin with a recent observation of Rao and Daumé III (2018), who considered the task of question reranking: a good clarification question is the one whose answer has a high *utility*, which they defined as the likelihood that this question would lead to an answer that will make the context more complete (§ 5.2.3). Inspired by this, we construct a question generation model that first generates a question given a context, and then generates a hypothetical answer to that

[1]For instance, under home appliances, frequently asking "Is it made in China?" or "What are the dimensions?"

question. Given this (context, question, answer) triple, we train a utility calculator to estimate the usefulness of this question. We then show that this utility calculator can be generalized using ideas for generative adversarial networks Goodfellow et al. (2014) for text Yu et al. (2017), wherein the utility calculator plays the role of the "discriminator" and the question generator is the "generator" (§ 5.2.2), which we train using the MIXER algorithm Ranzato et al. (2015).

We evaluate our approach on two question generation datasets. The first is the Stack Exchange dataset (Table 5.2) where given a post, we train a model to generate a clarification question that points at missing information that could be potentially useful to someone trying to resolve the issue in the post. The second is the Amazon dataset (Table 5.1) where given a product description, we train a model to generate a clarification question that points at missing information that a potentially buyer might find useful. Using both automatic metrics and human evaluation, we demonstrate that although all models generate questions that are relevant to the context at hand, our adversarially-trained model generates more useful and specific questions than all the baseline models.

5.2 Training a Clarification Question Generator

Our goal is to build a model that, given a context, can generate an appropriate clarification question. Our dataset consists of (*context*, *question*, *answer*) triples where the *context* is an initial textual context, *question* is the clarification question that asks about some missing information in the context and *answer* is the

Product title	T-fal Nonstick Cookware Set, 18 pieces, Red
Product description	Easy non-stick 18pc set includes every piece for your everyday meals. Exceptionally durable dishwasher safe cookware for easy clean up. Durable non-stick interior. Oven safe up to 350.F/177.C
Question	Are they induction compatible?
Answer	They are aluminium so the answer is NO.

Table 5.1: Sample product description from amazon.com paired with a clarification question and answer.

Title	Wifi keeps dropping on 5Ghz network
Post	Recently my wireless has been very iffy at my university. I notice that I am connected to a 5Ghz network, while I am usually connected to a 2.4Ghz everywhere else (where everything works just fine). Sometimes it reconnects, but often I have to run 'sudo service network-manager restart'. Is it possible a kernel update has caused this?
Question	what is the make of your wifi card ?
Answer	intel corporation wireless 7260 (rev 73)

Table 5.2: Sample post from stackexchange.com paired with a clarification question and answer.

answer to the clarification question (details in **??**). Representationally, our question generator is a standard sequence-to-sequence model with attention (§ 5.2.1). The learning problem is: how to train the sequence-to-sequence model to generate good clarification questions.

An overview of our training setup is shown in Figure 5.1. Given a context, our question generator, which is a sequence-to-sequence model, outputs a question. In order to evaluate the usefulness of this question, we then have a second sequence-to-

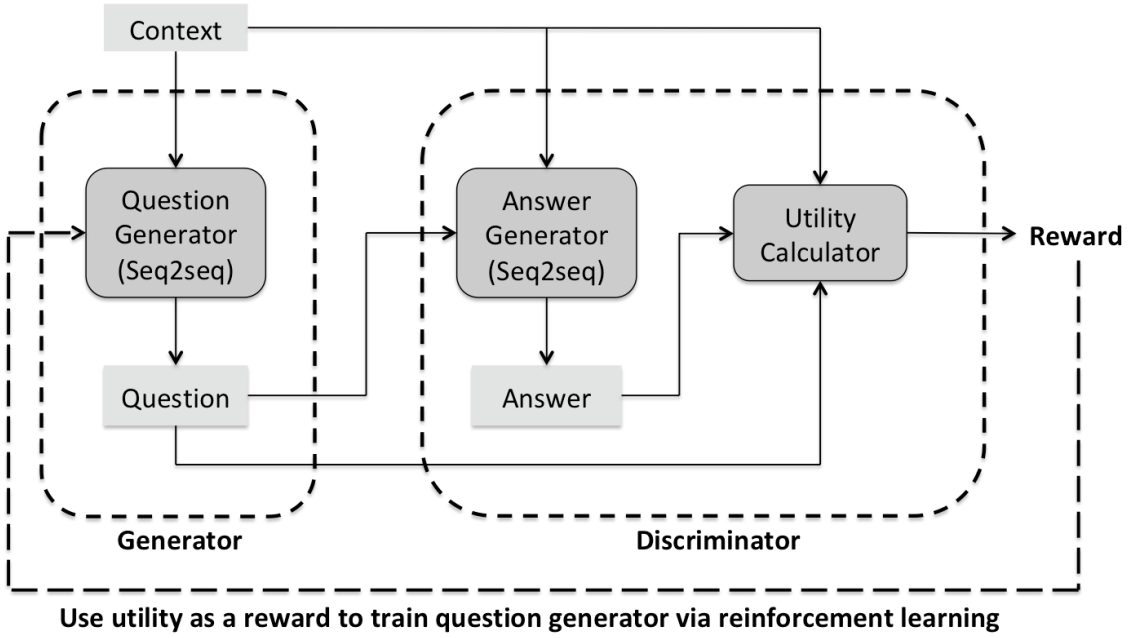

Figure 5.1: Overview of our GAN-based clarification question generation model.

sequence model called the "answer generator" that generates a hypothetical answer based on the context and the question (§ 5.2.5). This (context, generated question and generated answer) triple is fed into a UTILITY calculator, whose initial goal is to estimate the probability that this (question, answer) pair is useful in this context (§ 5.2.3). This UTILITY is treated as a reward, which is used to update the question generator using the MIXER Ranzato et al. (2015) algorithm (§ 5.2.2). Finally, we reinterpret the answer-generator-plus-utility-calculator component as a *discriminator* for differentiating between (context, true question, generated answer) triples and (context, generated question, generated answer) triples , and optimize the generator for this adversarial objective using MIXER (§ 5.2.4).

5.2.1 Sequence-to-sequence Model for Question Generation

We use a standard attention based sequence-to-sequence model (Luong et al., 2015) for our question generator. Given an input sequence (context) $c = (c_1, c_2, ..., c_N)$, this model generates an output sequence (question) $q = (q_1, q_2, ..., q_T)$. The architecture of this model is an encoder-decoder with attention. The encoder is a recurrent neural network (RNN) operating over the input word embeddings to compute a source context representation $\tilde{c}$. The decoder uses this source representation to generate the target sequence one word at a time:

$$p(q|\tilde{c}_t) = \prod_{t=1}^{T} p(q_t|q_1, q_2, ..., q_{t-1}, \tilde{c}_t) = \prod_{t=1}^{T} softmax(W_s\tilde{h}_t) \quad ; \quad \text{where } \tilde{h}_t = \tanh(W_c[\tilde{c}_t; h_t]) \tag{5.1}$$

In Eq 5.1, $\tilde{h}_t$ is the attentional hidden state of the RNN at time t and W_s and W_c are parameters of the model. The predicted token q_t is the token in the vocabulary that is assigned the highest probability using the softmax function. The standard training objective for sequence-to-sequence model is to maximize the log-likelihood of all (c, q) pairs in the training data D which is equivalent to minimizing the loss,

$$L_{\text{mle}}(D) = -\sum_{(c,q)\in D}\sum_{t=1}^{T} \log p(q_t|q_1, q_2, ..., q_{t-1}, c) \tag{5.2}$$

In Eq 5.1, $\tilde{h}_t$ is the attentional hidden state of the RNN at time t obtained by concatenating the target hidden state h_t and the source-side context vector $\tilde{c}_t$, and W_s is a linear transformation that maps h_t to an output vocabulary-sized vector. The predicted token q_t is the token in the vocabulary that is assigned the highest probability using the softmax function. Each attentional hidden state $\tilde{h}_t$ depends

on a distinct input context vector $\tilde{c}_t$ computed using a global attention mechanism over the input hidden states as:

$$\tilde{c}_t = \sum_{n=1}^{N} a_{nt} h_n \tag{5.3}$$

$$a_{nt} = \text{align}(h_n, h_t) = \exp\left[h_t^T W_a h_n\right] \Big/ \sum_{n'} \exp\left[h_t^T W_a h_{n'}\right] \tag{5.4}$$

The attention weights a_{nt} are calculated based on the alignment score between the source hidden state h_n and the current target hidden state h_t.

5.2.2 Training the Generator to Optimize Utility

Training sequence-to-sequence models for the task of clarification question generation (with context as input and question as output) using maximum likelihood objective unfortunately leads to the generation of highly generic questions, such as *"What are the dimensions?"* when asking questions about home appliances. Recently, Rao and Daumé III (2018) observed that the usefulness of a question can be better measured as the *utility* that would be obtained if the context were updated with the answer to the proposed question. Following this observation, we first use a pretrained answer generator (§ 5.2.5) to generate an answer given a context and a question. We then use a pretrained Utility calculator (§ 5.2.3) to predict the likelihood that the generated answer would increase the utility of the context by adding useful information to it. Finally, we train our question generator to optimize this Utility based reward.

Similar to optimizing metrics like Bleu and Rouge, this Utility calculator also operates on discrete text outputs, which makes optimization difficult due to non-

differentiability. A successful recent approach dealing with the non-differentiability while also retaining some advantages of maximum likelihood training is the Mixed Incremental Cross-Entropy Reinforce (Ranzato et al., 2015) algorithm (MIXER). In MIXER, the overall loss L is differentiated as in REINFORCE (Williams, 1992):

$$\begin{aligned} L(\theta) &= -\mathbb{E}_{q^s \sim p_\theta} r(q^s) \quad ; \\ \nabla_\theta L(\theta) &= -\mathbb{E}_{q^s \sim p_\theta} r(q^s) \nabla_\theta \log p_\theta(q^s) \end{aligned} \tag{5.5}$$

where y^s is a random output sample according to the model p_θ, where θ are the parameters of the network. The expected gradient is then approximated using a single sample $q^s = (q_1^s, q_2^s, ..., q_T^s)$ from the model distribution (p_θ). In REINFORCE, the policy is initialized randomly, which can cause long convergence times. To solve this, MIXER starts by optimizing maximum likelihood for the initial Δ time steps, and slowly shifts to optimizing the expected reward from Eq 5.5 for the remaining $(T - \Delta)$ time steps.

In our model, for the initial Δ time steps, we minimize L_{mle} and for the remaining steps, we minimize the following UTILITY-based loss:

$$L_{\text{max-utility}} = -(r(q^p) - r(q^b)) \sum_{t=1}^{T} \log p(q_t | q_1, ..., q_{t-1}, c_t) \tag{5.6}$$

where $r(q^p)$ is the UTILITY based reward on the predicted question and $r(q^b)$ is a baseline reward introduced to reduce the high variance otherwise observed when using REINFORCE. To estimate this baseline reward, we take the idea from the self-critical training approach Rennie et al. (2017) where the baseline is estimated using the reward obtained by the current model under greedy decoding during test time. We find that this approach for baseline estimation stabilizes our model better

than the approach used in MIXER.

5.2.3 Estimating a UTILITY Function from Historical Data

Given a (context, question, answer) triple, in the previous chapter we introduced a utility calculator UTILITY(c, q, a) to calculate the value of updating a context c with the answer a to a clarification question q. The inspiration for their utility calculator is to estimate the probability that an *answer* would be a meaningful addition to a context, and treat this as a binary classification problem where the positive instances are the true (context, question, answer) triples in the dataset whereas the negative instances are contexts paired with a random (question, answer) from the dataset. The model we use is to first embed the words in the context c, then use an LSTM (long-short term memory) (Hochreiter and Schmidhuber, 1997) to generate a neural representation $\bar{c}$ of the context by averaging the output of each of the hidden states. Similarly, we obtain a neural representation $\bar{q}$ and $\bar{a}$ of q and a respectively using question and answer LSTM models. Finally, a feed forward neural network $F_{\text{UTILITY}}(\bar{c}, \bar{q}, \bar{a})$ predicts the usefulness of the question.

5.2.4 UTILITY GAN for Clarification Question Generation

The UTILITY function trained on true vs random samples from real data (as described in the previous section) can be a weak reward signal for questions generated by a model due to the large discrepancy between the true data and the model's outputs. In order to strengthen the reward signal, we reinterpret the UTILITY

function (coupled with the answer generator) as a discriminator in an adversarial learning setting. That is, instead of taking the UTILITY calculator to be a fixed model that outputs the expected quality of a question/answer pair, we additionally optimize it to distinguish between true question/answer pairs and model-generated ones. This reinterpretation turns our model into a form of a generative adversarial network (GAN) (Goodfellow et al., 2014).

A GAN is a training procedure for "generative" models that can be interpreted as a game between a generator and a discriminator. The generator is an arbitrary model $g \in \mathcal{G}$ that produces outputs (in our case, questions). The discriminator is another model $d \in \mathcal{D}$ that attempts to classify between true outputs and model-generated outputs. The goal of the generator is to generate data such that it can fool the discriminator; the goal of the discriminator is to be able to successfully distinguish between real and generated data. In the process of trying to fool the discriminator, the generator produces data that is as close as possible to the real data distribution. Generically, the GAN objective is:

$$L_{\text{GAN}}(\mathcal{D}, \mathcal{G}) = \max_{d \in \mathcal{D}} \min_{g \in \mathcal{G}} \mathbb{E}_{x \sim \hat{p}} \log d(x) + \mathbb{E}_{z \sim p_z} \log(1 - d(g(z))) \tag{5.7}$$

where x is sampled from the true data distribution $\hat{p}$, and z is sampled from a prior defined on input noise variables p_z.

Although GANs have been successfully used for image tasks, training GANs for text generation is challenging due to the discrete nature of outputs in text. The discrete outputs from the generator make it difficult to pass the gradient update from the discriminator to the generator. Recently, Yu et al. (2017) proposed a

sequence GAN model for text generation to overcome this issue. They treat their generator as an agent and use the discriminator as a reward function to update the generative model using reinforcement learning techniques. By modeling the generator as a stochastic policy and directly training the policy via policy gradient, they avoid the differentiation difficulty at the cost of a much harder optimization problem. Our GAN-based approach is inspired by this sequence GAN model with two main modifications: a) We use the MIXER algorithm as our generator (§ 5.2.2) instead of policy gradient approach; and b) We use the UTILITY function (§ 5.2.3) as our discriminator instead of a convolutional neural network (CNN).

Theoretically, the discriminator should be trained using (context, true question, true answer) triples as positive instances and (context, generated question, generated answer) triples as the negative instances. However, we find that training a discriminator using such positive instances makes it very strong since the generator would have to not only generate real looking questions but also generate real looking answers to fool the discriminator. Since our main goal is question generation and since we use answers only as latent variables, we instead use (context, true question, *generated answer*) as our positive instances where we use the pretrained answer generator to get the *generated answer* for the true question. Formally, our objective function is:

$$L_{\text{GAN-U}}(\mathcal{U}, \mathcal{M}) = \max_{u \in \mathcal{U}} \min_{m \in \mathcal{M}} \mathbb{E}_{q \sim \hat{p}} \log u(c, q, \mathcal{A}(c, q)) + \mathbb{E}_{c \sim \hat{p}} \log(1 - u(c, m(c), \mathcal{A}(c, m(c)))) \tag{5.8}$$

where $\mathcal{U}$ is the UTILITY discriminator, $\mathcal{M}$ is the MIXER generator, $\hat{p}$ is our data of (context, question, answer) triples and $\mathcal{A}$ is our answer generator.

5.2.5 Pretraining

Question Generator. We pretrain our question generator using the sequence-to-sequence model (§5.2.1) to maximize the log-likelihood of all (context, question) pairs in the training data. Parameters of this model are updated during adversarial training.

Answer Generator. We pretrain our answer generator using the sequence-to-sequence model (§5.2.1) to maximize the log-likelihood of all ([context+question], answer) pairs in the training data. Parameters of this model are kept fixed during the adversarial training.[2]

Discriminator. In our UTILITY GAN model (§ 5.2.4), the discriminator is trained to differentiate between true and generated questions. However, since we want to guide our UTILITY based discriminator to also differentiate between true ("good") and random ("bad") questions, we pretrain our discriminator in the same way we trained our UTILITY calculator. For positive instances, we use a context and its true question, answer from the training data and for negative instances, we use the same context but randomly sample a question from the training data (and use the answer paired with that random question).

5.3 Experimental Results

We base our experimental design on the following research questions:

[2]We leave the experimentation of updating parameters of answer generator during adversarial training to future work.

1. Do generation models outperform simpler retrieval baselines?
2. Does optimizing the UTILITY reward improve over maximum likelihood training?
3. Does using adversarial training improve over optimizing the pretrained UTILITY?
4. How do the models perform when evaluated for nuances such as specificity and usefulness?

We evaluate our model on both the StackExchange and the Amazon datasets described in Chapter 3

5.3.1 Baselines and Ablated Models

We compare three variants (ablations) of our proposed approach, together with an information retrieval baseline:

GAN-Utility is our full model which is a UTILITY function based GAN training (§ 5.2.4) including the UTILITY discriminator, a MIXER question generator and a sequence-to-sequence based answer generator.

Max-Utility is our reinforcement learning baseline with a pretrained question generator described model (§ 5.2.2) without the adversarial training.

MLE is the question generator model pretrained on context, question pairs using

maximum likelihood objective (§5.2.1).

Lucene Given a context, we use Lucene to retrieve top 10 contexts that are most similar to the given context. We randomly choose a question from the 10 questions paired with these contexts to construct our Lucene baseline. For the Amazon dataset, we ignore questions asked to products of the same brand as the given product since Amazon replicates questions across same brand allowing the true question to be included in that set.

5.3.2 Experimental Details

In this section, we describe the details of our experimental setup. We preprocess all inputs (context, question and answers) using tokenization and lowercasing. We set the max length of context to be 100, question to be 20 and answer to be 20. We test with context length 150 and 200 and find that the automatic metric results are similar as that of context length 100 but the experiments take much longer. Hence, we set the max context length to be 100 for all our experiments. Similarity, we find that an increased length of question and answer yields similar results with increased experimentation time.

Our sequence-to-sequence model (§5.2.1) operates on word embeddings which are pretrained on in domain data using Glove (Pennington et al., 2014). As frequently used in previous work on neural network modeling, we use an embeddings of size 200 and a vocabulary with cut off frequency set to 10. During train time,

we use teacher forcing (Williams and Zipser, 1989). During test time, we use beam search decoding with beam size 5. We use a hidden layer of size two for both the encoder and decoder recurrent neural network models with size of hidden unit set to 100. We use a dropout of 0.5 and learning ratio of 0.0001. We use a batch size of 128.

In the MIXER model, we start with $\Delta = T$ and decrease it by 2 for every epoch (we found decreasing Δ to 0 is ineffective for our task, hence we stop at 2). We run the pretrain the question generator and the answer generator for 100 epochs and run the REINFORCE and the adversarial training for 8 epochs.

We would like to note here that our decisions of these hyperparameter settings have been influenced by the following previous works that have done a more systematic investigation of how these hyperparameters influence model predictions. Neishi et al. (2017) perform a detailed analysis of hyperparameter tuning of sequence-to-sequence models for the task of machine translation. Khandelwal et al. (2018) discuss how neural language models make use of context and find that these models are more sensitive to nearby contexts (upto 100 tokens) and less sensitive to tokens beyond that window. Qi et al. (2018) investigate the usefulness of using pretrained word embeddings and find that in case of scarcity of in-domain data (such as low resource machine translation), the use of pretrained word embeddings can be very effective.

5.3.3 Evaluation Metrics

We evaluate initially with several automated evaluation metrics, and then more substantially based on crowdsourced human judgments.

5.3.3.1 Automatic Metrics

Diversity, which calculates the proportion of unique trigrams in the output to measure the diversity as commonly used to evaluate dialogue generation (Li et al., 2016b). We report trigrams, but bigrams and unigrams follow similar trends.

Bleu (Papineni et al., 2002), which evaluates n-gram precision between a predicted sentence and reference sentences.

Meteor (Banerjee and Lavie, 2005), which is similar to BLEU but includes stemmed and synonym matches when measuring the similarity between the predicted sequence and the reference sequences.

5.3.3.2 Human Judgements

We use Figure-Eight (`https://www.figure-eight.com`), which is a crowdsourcing platform, to collect human judgements. Each question was annotated by five annotators. We paid crowdworkers 5 cents per judgment. Below are the exact wordings of the questions we asked the annotators with the numeric scores corresponding to each option:

Relevance: We ask *"Is the question on topic"* and let workers choose from:

1: Yes

0: No

Grammaticality: We ask *"Is the question grammatical?"*, and let workers choose from:

1: Yes

0: No

Seeking new information: We ask *"Does the question ask for new information currently not included in the description?"* and let workers choose from:

1: Yes

0: No

Specificity: We ask *"How specific is the question?"* and let workers choose from:

4: Specific pretty much only to this product (or same product from different manufacturer)

3: Specific to this and other very similar products

2: Generic enough to be applicable to many other products of this type

1: Generic enough to be applicable to any product under Home and Kitchen

N/A (Not applicable): Question is not on topic OR is incomprehensible

Usefulness: We ask *"How useful is the question to a potential buyer (or a current user) of the product?"* and let workers choose from:

4: Useful enough to be included in the product description

3: Useful to a large number of potential buyers (or current users)

2: Useful to a small number of potential buyers (or current users)

1: Useful only to the person asking the question

Criteria	Agreement
Relevance	0.92
Grammaticality	0.92
Seeking new information	0.84
Usefulness	0.65
Specificity	0.72

Table 5.3: Inter-annotator agreement on the five criteria used in human-based evaluation.

N/A (Not applicable): Question is not on topic OR is incomprehensible OR is not seeking new information

Since the inter-annotator agreement on the usefulness criteria was low (refer to Table 5.3), in order to reduce the subjectivity involved in the fine grained annotation, we convert the range [1-4] to a more coarse binary range [0-1] by mapping the scores 4 and 3 to **1** and the scores 2 and 1 to **0**.

The inter annotator agreement on each of the above five criteria is shown in Table 5.3. Agreement on Relevance, Grammaticality and Seeking new information is high. This is not surprising given that these criteria are not very subjective. On the other hand, the agreement on usefulness and specificity is quite moderate since these judgments can be very subjective.

	Amazon			StackExchange		
Model	Diversity	Bleu	Meteor	Diversity	Bleu	Meteor
Reference	0.6934	—	—	0.7509	—	—
Lucene	0.6289	4.26	10.85	0.7453	1.63	7.96
MLE	0.1059	**17.02**	12.72	0.2183	3.49	8.49
Max-Utility	0.1214	16.77	12.69	**0.2508**	3.89	8.79
GAN-Utility	**0.1296**	15.20	**12.82**	0.2256	**4.26**	**8.99**

Table 5.4: Diversity as measured by the proportion of unique trigrams in model outputs. Bleu and Meteor scores using up to 10 references for the Amazon dataset and up to six references for the StackExchange dataset. Numbers in bold are the highest among the models. All results for Amazon are on the entire test set whereas for StackExchange they are on the 500 instances of the test set that have multiple references.

5.3.4 Automatic Metric Results

Table 5.4 shows the results on the two datasets when evaluated according to automatic metrics.

In the Amazon dataset, GAN-Utility outperforms both MLE and Max-Utility models on Diversity, suggesting that it produces more diverse outputs. Lucene, on the other hand, has the highest Diversity since it consists of human generated questions, which tend to be more diverse because they are much longer compared to

model generated questions. This comes at the cost of lower match with the reference as visible in the BLEU and METEOR scores. In terms of BLEU and METEOR, there is inconsistency. Although GAN-Utility outperforms all baselines according to METEOR, the fully ablated MLE model has a higher BLEU score. This is because BLEU score looks for exact n-gram matches and since MLE produces more generic outputs, it is much more likely that it will match one of 10 references compared to the specific/diverse outputs of GAN-Utility, since one of those ten is highly likely to itself be generic.

In the StackExchange dataset GAN-Utility outperforms both MLE and Max-Utility models on both BLEU and METEOR. Unlike in the Amazon dataset, MLE does not outperform GAN-Utility in BLEU. This is because the MLE outputs in this dataset are not as generic as in the Amazon dataset due to the highly technical nature of contexts in StackExchange. As in the Amazon dataset, GAN-Utility outperforms MLE on DIVERSITY. Interestingly, the Max-Utility ablation achieves a higher DIVERSITY score than GAN-Utility. On manual analysis we find that Max-Utility produces longer outputs compared to GAN-Utility but at the cost of being less grammatical.

5.3.5 Human Judgements Analysis

Table 5.5 shows the numeric results of human-based evaluation performed on the reference and the system outputs on 300 random samples from the test set

Model	Relevant [0-1]	Grammatical [0-1]	New Info [0-1]	Useful [0-1]	Specific [0-4]
Reference	0.96	0.99	0.93	0.72	3.38
Lucene	**0.90**	**0.99**	**0.95**	0.68	2.87
MLE	**0.92**	**0.96**	0.85	0.91	3.05
Max-Utility	**0.93**	**0.96**	0.88	0.91	3.29
GAN-Utility	**0.94**	**0.96**	0.87	**0.96**	**3.52**

Table 5.5: Results of human judgments on model generated questions on 300 sample Home & Kitchen product descriptions. The options described in § 5.3.3 are converted to corresponding numeric range (see supplementary material). The difference between the bold and the non-bold numbers is statistically significant with $p < 0.05$. Reference is excluded in the significance calculation.

of the Amazon dataset.[3] Overall, these results show that the GAN-Utility model successfully generates the most useful and the most specific questions while being equally good at seeking new information. All approaches produce relevant and grammatical questions. All models are all equally good at seeking new information, but are weaker than Lucene, which performs better at seeking new information but at the cost of much lower specificity and lower usefulness.

Our full model, GAN-Utility, performs significantly better at the usefulness criteria showing that the adversarial training approach generates more useful ques-

[3]We could not ask crowdworkers evaluate the StackExchange data due to its highly technical nature.

tions. Interestingly, all our models produce questions that are more useful than Lucene and Reference, largely because Lucene and Reference tend to ask questions that are more often useful only to the person asking the question, making them less useful for potential other buyers (see Figure 5.3). GAN-Utility also performs significantly better at generating questions that are more specific to the product (see details in Figure 5.2), which aligns with the higher DIVERSITY score obtained by GAN-Utility under automatic metric evaluation.

Table 5.6 contains example outputs from different models along with their usefulness and specificity scores. MLE generates questions such as *"is it waterproof?"* and *"what is the wattage?"*, which are applicable to many other products. Whereas our GAN-Utility model generates more specific question such as *"is this shower curtain mildew resistant?"*. We provide further analysis of system outputs on both Amazon and Stack Exchange datasets in the next section.

5.3.6 Analysis of System Outputs on Amazon Dataset

Table 5.7 shows the system generated questions for three product descriptions in the Amazon dataset.

In the first example, the product is a shower curtain. The Reference question is specific and highly useful. Lucene, on the other hand, picks a moderately specific ("how to clean it?") but useful question. MLE model generates a generic but useful "is it waterproof?". Max-Utility generates comparatively a much longer question but in doing so loses out on relevance. This behavior of generating two unrelated

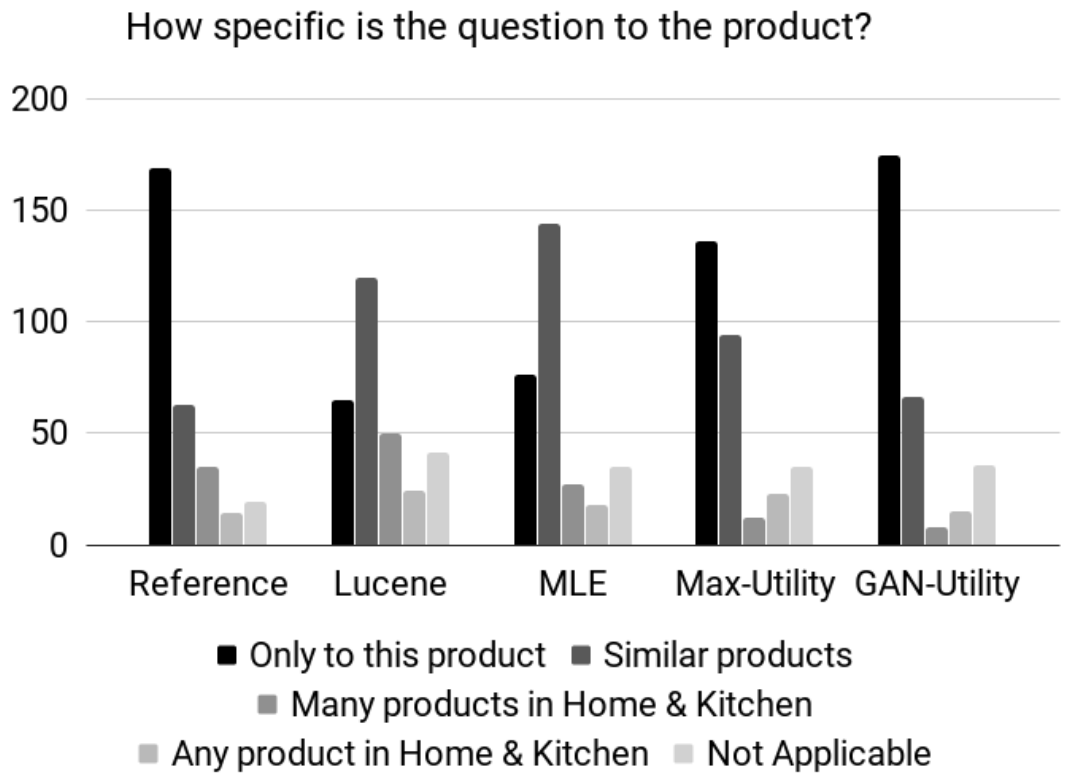

Figure 5.2: Results of human judgements on the specificity criteria.

sentences is observed quite a few times in both Max-Utility and GAN-Utility models. This suggests that these models, in trying to be very specific, end up losing out on relevance. In the same example, GAN-Utility also generates a fairly long question which, although awkwardly phrase, is quite specific and useful.

In the second example, the product is a Duvet Cover Set. Both Reference and Lucene questions here are examples of questions that are pretty much useful only to the person asking the question. We find many such questions in both Reference and Lucene outputs which is the main reason for the comparatively lower usefulness scores for their outputs. All three of our models generate irrelevant questions since

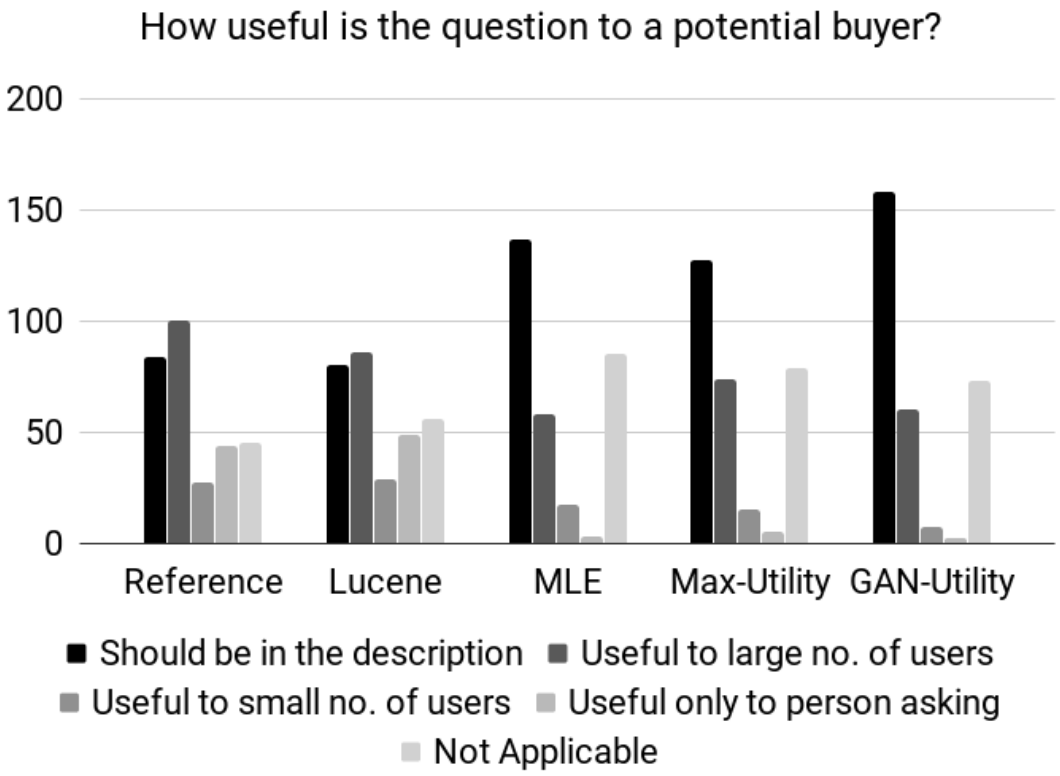

Figure 5.3: Results of human judgements on the usefulness criteria.

the product description explicitly says that the set is full size.

In the last example, the product is a set of mopping clothes. Reference question is quite specific but has low usefulness. Lucene picks an irrelevant question. MLE and Max-Utility generate highly specific and useful questions. GAN-Utility generates an ungrammatical question by repeating the last word many times. We observe this behavior quite a few times in the outputs of both Max-Utility and GAN-Utility models suggesting that our sequence-to-sequence models are not very good at maintaining long range dependencies.

Title	Raining Cats and Dogs Vinyl Bathroom **Shower Curtain**		
Product Description	This adorable shower curtain measures 70x72 inches and would make a great gift!		
		Useful [1-4]	Specific [1-4]
Reference	does the vinyl smells?	3	4
Lucene	other than home sweet home , what other sayings on the shower curtain ?	2	4
MLE	is it waterproof ?	4	2
Max-Utility	is this shower curtain mildew ?	N/A	N/A
GAN-Utility	is this shower curtain mildew resistant ?	4	4
Title	PURSONIC HF200 Pedestal **Bladeless Fan & Humidifier** All-in-one		
Product Description	The first bladeless fan to incoporate a humidifier! , This product operates solely as a fan, a humidifier or both simultaneously. 5.5L tank lasts up to 12 hours.		
		Useful [1-4]	Specific [1-4]
Reference	i can not get the humidifier to work	1	2
Lucene	does it come with the vent kit	3	3
MLE	what is the wattage of this fan ?	4	2
Max-Utility	is this battery operated ?	3	2
GAN-Utility	does this fan have an automatic shut off ?	4	4

Table 5.6: Example outputs from each of the systems for two product descriptions along with the usefulness and the specificity score given by human annotators. Descriptions of scores are in the supplementary material.

5.3.7 Analysis of System Outputs on Stack Exchange Dataset

Table 5.8 includes system outputs for three posts from the Stack Exchange dataset.

The first example is of a post where someone describes their issue of not being able to recover from their boot. Reference and Lucene questions are useful. MLE generates a generic question that is not very useful. Max-Utility generates a useful question but has slight ungrammaticality in it. GAN-Utility, on the other hand, generates a specific and an useful question.

In the second example, again Reference and Lucene questions are useful. MLE generates a generic question. Max-Utility and GAN-Utility both generate fairly specific question but contain unknown tokens. The Stack Exchange dataset contains several technical terms leading to a long tail in the vocabulary. Owing to this, we find that both Max-Utility and GAN-Utility models generate many instances of questions with unknown tokens.

In the third example, the Reference question is very generic. Lucene asks a relevant question. MLE again generates a generic question. Both Max-Utility and GAN-Utility generate specific and relevant questions.

5.4 Conclusion

In this chapter, we describe a novel approach to the problem of clarification question generation. Given a context, we use the observation from the previous chapter that the usefulness of a clarification question can be measured by the value of updating the context with an answer to the question. We use a sequence-to-sequence model to generate a question given a context and a second sequence-to-sequence model to generate an answer given the context and the question. Given the (context, predicted question, predicted answer) triple we calculator the utility of this triple and use it as a reward to retrain the question generator using reinforcement learning based MIXER model. Further, to improve upon the utility function, we reinterpret it as a discriminator in an adversarial setting and train both the utility function and the MIXER model in a minimax fashion. We find that our adversar-

ial training approach produces more diverse questions compared to both a model trained using maximum likelihood objective and a model trained using utility reward based reinforcement learning. A model that can generate diverse questions is useful since a model that only generates generic questions can be very repetitive and might not help elicit useful information about a given context. Motivated by this idea, in the next chapter, we look into how can we guide our clarification question generation model to generate more specific questions.

Title	Chezmoi Collection 7-piece Chic Ruched Duvet Cover Set, Full Size (with Pillows)		
Product Description	Create a world of romance with the elegant, and luxurious all white duvet cover set.		
		Useful [1-4]	Specific [1-4]
Reference	How long will it take to ship this to my pin code?	1	1
Lucene	and can you use the duvet as is ? if not what shall i purchase to put in it for winter or fall ?	1	4
MLE	what are the dimensions of the king size ?	N/A	N/A
Max-Utility	what are the dimensions of the king size ?	N/A	N/A
GAN-Utility	does the king size come with a duvet cover or do you have to buy a king duvet ?	N/A	N/A
Title	Microfiber 3-Pack, Pro-Clean Mopping Cloths for Braava Floor Mopping Robot		
Product Description	Braavas textured Pro-Clean microfiber mopping cloths remove dirt and hair from your floors. The cloths can be washed and used hundreds of times They are compatible with all Braava models, including the Pro-Clean Reservoir Pad. Each cloth is easy to attach and remove from the magnetic cleaning pad.		
		Useful [1-4]	Specific [1-4]
Reference	do i have to use a new cloth every time i want to clean my floor? $5/$6 seems expensive per clean	2	4
Lucene	do they remove pet odor ?	N/A	N/A
MLE	will these work with the scooba ?	3	3
Max-Utility	do these cloths work on hardwood floors ?	3	4
GAN-Utility	will this work with the scooba mop mop mop mop mop mop	N/A	N/A

Table 5.7: Example outputs from each of the systems for three product descriptions from the Home & Kitchen category of the Amazon dataset.

Title	how can i recover my boot ?
Post	since last week i am trying to recover my boot . after the last update for ubuntu 12.04 i lost it and nobody could help me , i used boot-repair but there are problems with dependences , which ca n't be fix . i will be very grateful if somebody could help me .
Reference	what happens when you try to boot ?
Lucene	can you get into bios ?
MLE	how much ram do you have ?
Max-Utility	do you have a swap partition partition ?
GAN-Utility	what happens when you try to boot into safe mode ?
Title	packages have unmet dependencies when trying to install anything
Post	i 'm running ubuntu 14.04 lts . ive recently run into this problem when updating or installing , after trying a few solutions to no avail , but now i 'm having the same issue with steam trying to update , which i use quite a lot . ive looked through dozens of posts about similar issues and tried a lot of solutions and nothing seems to work.
Reference	sudo dpkg -reconfigure all ? ?
Lucene	if you use the graphical package manager , does n't add the required packages automatically ?
MLE	how long did you wait ?
Max-Utility	can you post the output of 'apt-cache policy UNK ?
GAN-Utility	can you post a screenshot of the output of 'sudo apt-get install UNK
Title	full lubuntu installation on usb (uefi capable)
Post	i want to do a full lubuntu installation on a usb stick that can be booted in uefi mode. i do not want persistent live usb but a full lubuntu installation (which happens to live on a usb stick) and that can boot fromanyuefi-capable computer ...
Reference	hello and welcome on askubuntu . could you please clarify what you want ?
Lucene	so , ubuntu was installed to the pen drive ?
MLE	which version of ubuntu ?
Max-Utility	do you have a live cd or usb stick ?
GAN-Utility	what is the model of the usb stick ?

Table 5.8: Example outputs from each of the systems for three posts of the Stack Exchange dataset.

Chapter 6: Specificity-Controlled Question Generation Model

6.1 Introduction

In the last chapter, we saw how we can train a sequence-to-sequence neural network model to generate a useful question given an under-specified context. We used answer-based adversarial training strategy to train the sequence-to-sequence model. One of our key findings was that an adversarially trained model generates questions that are more specific to the context compared to a model trained using the traditional maximum-likelihood training objective. Generating questions with a desired level of specificity can be useful in many scenarios. For instance, consider an automated agent assisting a human in a technical issue through a dialogue. At the start of the conversation, we would want the automated agent to ask the human more generic questions in order to understand the general domain of the problem. Whereas, at a later stage of the conversation, we would want the agent to ask more specific questions to narrow down the problem. In the e-retail scenario considered in this dissertation, if the given description belongs to a product which is similar to several other products that currently exist in the dataset, then we might want our automated system to generate more specific questions (since we could easily generate generic questions for this product by retrieving the top-K

frequently asked questions in the dataset, for instance). On the other hand, if the given product belongs to a fairly new category, then we might want our system to generate more generic questions. In this chapter, therefore, we propose to build a model that given a context and a level of specificity (specific or generic), generates a question with that level of specificity. For instance, in Figure 6.1, given a product description (context) and a level of specificity as "<generic>", our goal is to generate a question such as *"Where was this manufactured?"* which is applicable to many products on amazon.com. Whereas, given the same product description and the level of specificity as "<specific>", we would like to generate a question that is more specific to the given product such as *"Is this induction safe?"*

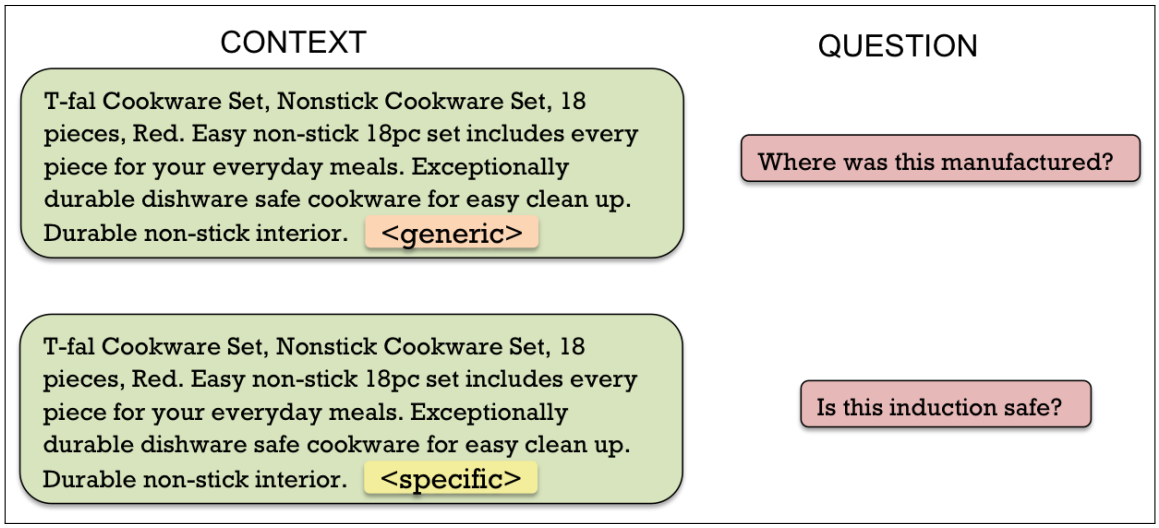

Figure 6.1: Sample product description from amazon.com paired with a generic and a specific clarification question.

We take a semi-supervised approach to our problem of generating specificity controlled questions. Motivated by Sennrich et al. (2016), we build a question generation model that incorporates the level of specificity as additional input signal

during training[1]. In our work, we hypothesize that at training time if we append the context (source) with the level of specificity of the question (target), then the model will learn how to generate questions that at a given level of specificity. In Figure 6.2, the question generation model is trained using context appended with specificity as input and question as the output. In order to do this training, we would need to label all the questions in our training data with their level of specificity i.e. generic vs specific. Doing this labeling manually for the entire training dataset of approximately 150K questions would be too expensive. Hence, we train a supervised model that automatically labels a question (given a context) with its level of specificity to the given context. Figure 6.2 shows our specificity classifier trained using a relatively small set of questions manually annotated with their level of specificity.

Our specificity classifier is inspired by the model introduced by Louis and Nenkova (2011) who train a binary classifier to automatically identify generic vs specific sentences in news articles. Their classifier is based on features that capture lexical and syntactic information, as well as specificity and word polarity. They use human annotators to manually annotate a set of sentences with generic/specific labels and train a binary classifier using a logistic regression model. Following their work, we use crowdsourcing to annotate a set of 3000 questions from the Amazon dataset with their level of specificity to the product description. We use this annotated data to train a binary classifier to predict the level of specificity of a question, given a context. We use some of the features introduced by Louis and

[1]Sennrich et al. (2016) refer to this as side constraints.

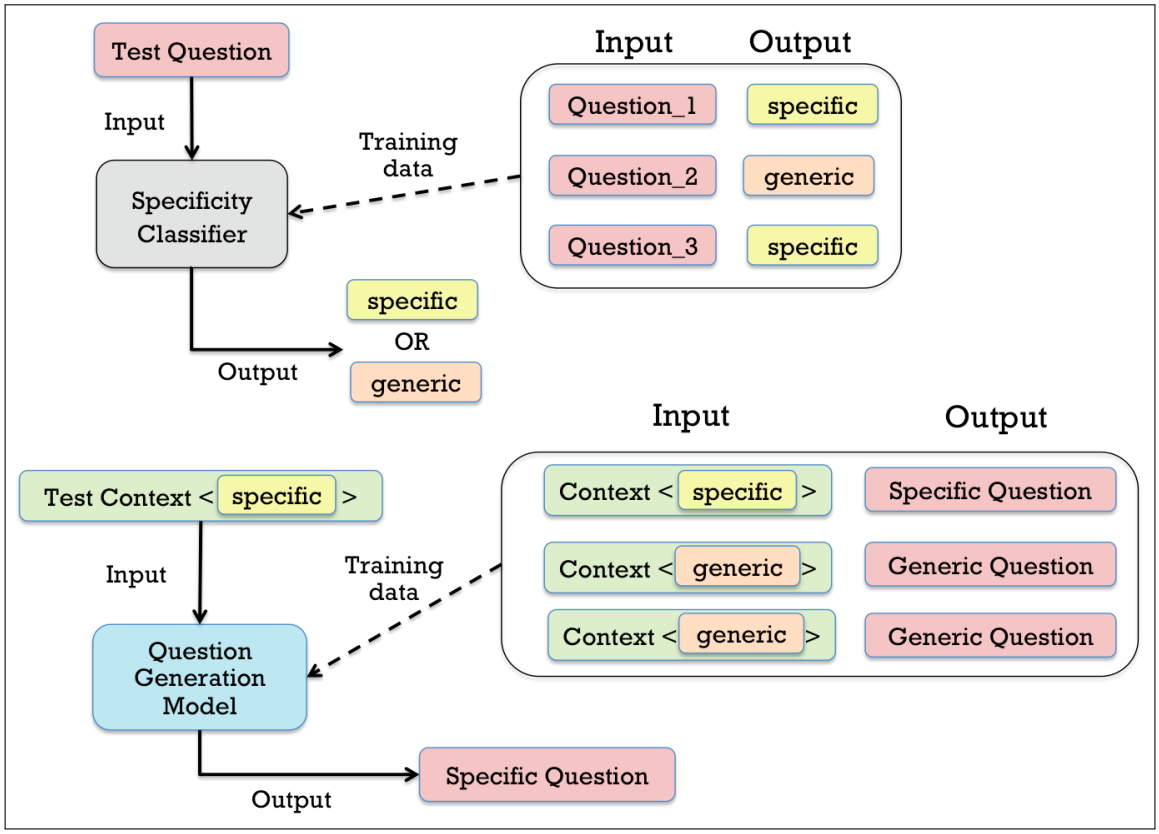

Figure 6.2: Specificity-controlled question generation model.

Nenkova (2011) and introduce new features that are indicative of the specificity level of the question to train our binary classifier.

We use our specificity classifier to append the context with the level of specificity of the target question. We finally retrain the question generation model described in the previous chapter with the modified context. At test time, given a context appended with a level of specificity (generic or specific), our model generates a clarification question at that level of specificity.

6.2 Related Work

We consider specificity as a dimension of style. Sociolinguistics defines style as a set of linguistic variants with specific social meanings. Hovy (1987) argues that by varying the style of a text, people convey more information than is present in the literal meaning of the words. In order to build automated intelligent agents that can effectively communicate with humans, it is important that we teach these agents to recognize the various stylistic variations in human language and also teach them to generate language in a particular given style. In the field of natural language processing, there has been previous work on both identifying style and generating text in a given style.

Under style identification, there has been work on detecting formality of a given text at the lexical level (Brooke and Hirst, 2014; Brooke et al., 2010; Lahiri et al., 2011; Pavlick and Nenkova, 2015), at the sentence level (Pavlick and Tetreault, 2016) and at the document level (Mosquera and Moreda, 2012; Peterson et al., 2011; Sheikha and Inkpen, 2010). Markowitz and Hancock (2016) studied writing styles in fraudulent papers whereas Feng et al. (2012) build models for deception detection. Koppel et al. (2002, 2009, 2011) develop machine learning models for authorship identification, where the style corresponds to the writing style of an author. Previous work most relevant to us is the work around detecting generic/specific distinctions of text. Reiter and Frank (2010) introduce a method for distinguishing between noun phrases that describes class of individuals (generic) versus those that refer to specific individuals. Mathew (2009) distinguish between sentences that relate to

specific event versus those that relate to general facts. Louis and Nenkova (2011) build a model to automatically identify general and specific sentences motivated by potential applications in summarization and writing feedback.

Generating style-controlled text has been studied in three different settings before: supervised learning, semi-supervised setting and unsupervised setting. Under supervised setting, Xu et al. (2012) develop a statistical machine translation based model for paraphrasing sentences into Shakespearean English whereas Jhamtani et al. (2017) develop a neural machine translation based model for the same task. Recently, we (Rao and Tetreault, 2018) developed models for automatically rewriting sentences from informal to formal style and vice-versa. Under semi-supervised setting, Sennrich et al. (2016) develop models to control politeness of the generated text using side constraints where the source is appended with an artificial token denoting the style in which we want the model to generate its target. Yamagishi et al. (2016) use a similar idea for controlling the voice of the generated text. Niu et al. (2017, 2018) control formality during translation. Under unsupervised setting, Hu et al. (2017) control the sentiment and the tense of the generated text by learning a disentangled latent representation in a neural generative model. Ficler and Goldberg (2017) control several linguistic style aspects simultaneously by conditioning a recurrent neural network language model on specific style (professional, personal, length) and content (theme, sentiment) parameters.

6.3 Annotating Questions with Specificity Level

The key idea behind the use of side constraints is to guide a model to generate text constrained with a certain linguistic phenomenon by training it on sentences that have been annotated with such constraints. In our scenario, the constraint is the level of specificity. More specifically, our input is the context and the output is the question as the per the specified level of specificity. Hence, while training this model we need to append the source i.e. the context with the level of specificity of the target i.e. the question. Given that our neural network based question generation model requires huge amounts of training data, annotating the entire training data (around 100K questions) with the level of specificity manually would be too time consuming and costly. Therefore, we take a machine learning approach to this problem where we annotate a subset of the training data using humans and train a machine learning model on this annotated data which learns to predict the level of specificity of a question given the context. In this section, we describe how we collect human annotations on the subset of the training data.

6.3.1 Annotation Design

We define our annotation task as given a context and a clarification question, annotate if the question is generic or specific to the given context. One obvious way to do this task would be to show the annotators the context and the question and ask them to choose between generic or specific. However, we found that doing this annotation task for a question (given a context) without knowing the other

questions asked to that context is really hard and unintuitive. We found that an easier task would be to compare the level of specificity of two questions given a context. For instance, given the context in Figure 1.2, annotating the level of specificity of the question "Are they ok for induction stove?" in isolation is difficult. However, comparing the specificity level of this question with say another question "Where are they made?" is easier, we can say that the former question is more specific than the latter since the latter is applicable to a larger set of products. Hence, we design an annotation scheme where given a context and two questions Question A and Question B, we ask annotators to compare the level of specificity of the two questions by choosing from the following options:

1. Question A is more specific
2. Question B is more specific
3. Both questions are at the same level of specificity

Each question pair is annotated by five annotators. We use Figure-Eight to collect these annotations. Each pair of questions is annotated by five annotators.[2]

6.3.2 Getting Specificity Levels from Annotations

The next step would be how to convert these comparisons into individual generic/specific labels for the questions. Given a context and the N questions asked to that context, we collect annotations such that each question is compared to K other questions in the set N. Each question pair (q_i, q_j) is annotated by five an-

[2]We started with three annotators per pair of questions but obtained a low inter-annotator agreement and hence we moved up to five annotators.

notators. The platform we use to collect annotations assigns a trust value to each of its annotators based on the number of annotations performed by the annotator and how well the annotator performed on the test questions.[3] This trust value is between 0 and 1. We calculate the specificity score for each question as

$$\text{specificity\_score}(q_i) = \frac{1}{K}\sum_{j=1}^{K}\frac{1}{5}\sum_{a=1}^{5} t_a * d_a(i,j)$$

where t_a is the trust of the annotator a who annotated the question pair (q_i, q_j),
$d_a(i,j) = 1$; if annotator a annotated q_i as more specific than q_j,
$d_a(i,j) = -1$; if annotator a annotated q_j as more specific than q_i,
$d_a(i,j) = 0$; if annotator a annotated q_i is at the same level of specificity as q_j

The specificity score calculated as above is a value between -1 and 1. Given this value, we set a threshold S and when the score for a question is less than S, we label it as generic whereas when it is greater than or equal to S, we label it as specific. We set a global threshold of $S = 0$ for all contexts.

If we collect annotations such that each question is compared to every other questions in the set of N, then we could get a more accurate specificity score for a question. However, given that N can be as high as 10, collecting $\frac{N(N-1)}{2}$ annotations per context could be expensive. We, therefore, collect annotations such that each question is compared to two other questions in the set N.

To ensure that this method is reliable, for 25 of the contexts, we collect anno-

[3]The trust score assigned by the platform is similar to inter-annotator agreement.

tations such that each question is compared to every other question in the set N. On this subset, we calculate the specificity scores of the questions using $(N-1)$ comparisons per question ($S_{\text{all\_comparisons}}$) and we calculate the specificity scores of the questions using two comparisons per question ($S_{\text{two\_comparisons}}$). In order to understand how much do the specificity scores vary when they are calculated using these two different methods, we calculate the accuracy of the $S_{\text{two\_comparisons}}$ scores over the $S_{\text{all\_comparisons}}$ scores. We get an accuracy of 0.89 suggesting that, although the scores calculated using two comparisons can be noisy, they do not deviate too much from those obtained using all comparisons.

6.4 Model for Automatically Predicting Specificity Level

Given the specific/generic annotations on a subset of our training data, our next step is to train a machine learning model that can learn to predict the specificity level given a context and a question. Louis and Nenkova (2011) introduce a supervised classifier for automatically predicting whether a sentence in a summary is generic or specific. They define specificity as the level of detail present in a given sentence. The definition of specificity in our setting is how specific the question is to the given context. Their classifier is based on lexical and syntactic features. We use some of the features described in their work and introduce some new features relevant to our setting to create a similar classifier that predicts the level of specificity of a question given its context. The features used in our model are described below:

Question Length. Generic questions tend to be shorter in length compared to specific questions. For instance, "What are the dimensions?", "What is the size of the pillow?" are shorter in length compared to questions like "Does this pillow have a zipper or does it come with a cover?". We count the number of words in the question and use the count as a feature. Additionally, we use a part-of-speech tagger to tag the words in the question and count the number of nouns in the question and use that as feature. These two features were used by Louis and Nenkova (2011) in their model as well.

Path in WordNet. Questions that are more specific to a context tend to have more specific words. Motivated by this idea, we compute the length of the path of every noun and verb in a question to the root of WordNet (Miller et al., 1990) tree through hypernym relations. Longer paths would indicate that the words are more specific. Similar to Louis and Nenkova (2011), we use the average, min and max values of these lengths and use them as features.

Inverse Document Frequency. Another way to identify specific words is to calculate its inverse document frequency (IDF). IDF of a given term is defined by the inverse of the number of documents that contain that term. More formally $\text{IDF}(w) = \log(\frac{1}{\text{count of docs containing } w})$. In our setting, we consider a product description to be a document. So the IDF of a word in a given question is defined by the inverse of the counts of product descriptions that contain that word. We cal-

culate the IDF for every word in the question and include the maximum IDF, the minimum IDF and the average IDF as features. This feature is similar to the one used by Louis and Nenkova (2011) except that instead of calculating the document frequency over New York Times articles, we calculate the document frequency over product descriptions

Syntax. Similar to Louis and Nenkova (2011), we find that the use of nouns, adjectives and cardinals are good indicators of specificity. For instance, more specific questions tend to use more proper nouns, adjectives and cardinals (numbers). We use parts-of-speech tagger to tag the words in the questions and include the counts of proper nouns (NNP), adjectives (ADJ) and cardinals (CD) as features.

Polarity. Louis and Nenkova (2011) find that word polarity can be strong indicator of the level of specificity. For instance, strong opinions are indicative of generic sentences. To identify positive, negative and polar words, they use The General Inquirer and the MPQA Subjectivity lexicons. We find that these two lexicons, which mainly contain words frequently appearing in news articles, are less relevant for us due to the different nature of our dataset. Hence, we use the Linguistic and Word Inquiry (LIWC) (Pennebaker et al., 2001) instead.[4] We use the dictionary category of words in the question as features. Specifically, we consider the following categories under cognitive processes: *insight, causation, discrepancy, tentative, certainty, differentiation*. For each of these categories, we count the number of words

[4] `http://lit.eecs.umich.edu/~geoliwc/LIWC_Dictionary.htm`

in question that belong to that category and include that as a feature.

Question bag-of-words. We define a vector of the size of the vocabulary over the words in all the questions of our train set. Given a question, we set all the word positions that are included in the question to one in the vector and set the remaining to zero. We include this vector as a feature. This is similar to the "lexical (words)" features used by Louis and Nenkova (2011).

The features described above were adapted from Louis and Nenkova (2011). We now describe the new features we introduced specifically for our problem.

***Average word embeddings.** We train GloVe (Pennington et al., 2014), a word embedding model, on all contexts and questions in our Amazon dataset. We compute an average over the word embeddings of all the words in the question ($\bar{q}$) and include it as a feature. Likewise, we compute an average over the word embeddings of all the words in the context ($\bar{c}$) and include it as a feature.

***Similarity to context using word embeddings.** Louis and Nenkova (2011) define generic/specific based on the level of detail present in a sentence in isolation. In contrast, the specificity in our setting is measured by how specific is the question to the given context. Hence, we find that the similarity between the question and the given context to be a useful indicator of specificity. We measure this similarity using two ways. In the first way, we measure the similarity between the context and

the question in the vector semantic space. We compute an average over the word embeddings of all the words in the context ($\bar{c}$). Similarly, we compute an average over the word embeddings of all the words in the question ($\bar{q}$). We calculate the cosine similarity between $\bar{c}$ and $\bar{q}$ and use it as a feature.

***Similarity to context using WordNet.** In the second way, we measure the similarity in the WordNet space. Resnik (1995) compute semantic similarity between word pairs by looking at the minimal path between the words in WordNet. Motivated by this idea, we look at the hypernym relation path of every word in the question and every word in the context and count the number of hypernyms that were common in the two paths. We do this for every word pair (w_q, w_c) where w_q is a word in the question and w_c is the word in the context and use the aggregate count as a feature.

Given these features, we train a logistic regression model to make a binary prediction (-1: generic, 1: specific) given a context and a question. We use the Adam (**?**) optimizer. We use $L2$ regularizer.

6.5 Specificity-Controlled Question Generation Model

We use the specificity classifier described in the previous section to label all the questions in the training (and tune) data with generic/specific labels. We use these labels to append each context with the <*specific*> tag when the question paired

with the context is labeled as specific and with the <*generic*> tag when the question paired with the context is labeled as generic. We use this specificity annotated training data to train two specificity-controlled question generation model:

Specificity-MLE: Similar to the **MLE** model in the previous chapter, we train a sequence-to-sequence learning model (Sutskever et al., 2014) on (context+specificity, question) pairs using maximum likelihood objective (§ 5.2.1).

Specificity-GAN-Utility: This is the full question generation model described in previous chapter which we train using
(context+specificity, question) pairs instead of (context, question) pairs. We first pretrain a question generator on (context+specificity, question) pairs and an answer generator model using (context+specificity+question, answer) pairs using maximum likelihood objective. We then fine tune the question generator model using UTILITY function based GAN training (§ 5.2.4) including the UTILITY discriminator, a MIXER question generator.

At test time, we predict the specificity level of the target question using our specificity classifier and append the tag corresponding to that label to the context.

Features	Train Accuracy	Test Accuracy
Question length	0.55	0.55
Path in WordNet	0.63	0.64
Inverse Document Frequency	0.58	0.57
Syntax	0.71	0.70
Polarity	0.65	0.65
Question bag-of-words	0.80	0.71
*Average word embeddings	0.66	0.64
*Similarity to context using embeddings	0.58	0.59
*Similarity to context using WordNet	0.57	0.55
All features	0.79	0.73

Table 6.1: Average specificity classifier accuracy under 10 fold cross validation on train set and test set using different feature sets. * denotes new features not present in the model by Louis and Nenkova (2011).

6.6 Experimental Results

6.6.1 Specificity Classifier Results

We randomly select 500 contexts from our Amazon dataset and collect specificity annotations on the questions asked to those contexts. Given that each context has six questions on an average, we collect annotations on a total of 3310 questions. 2034 questions were annotated as generic and remaining were annotated as specific.

	Generic			Specific		
Model	Diversity	Bleu	Meteor	Diversity	Bleu	Meteor
Reference	0.6071	—	—	0.7474	—	—
Lucene	0.6289	2.90	12.04	0.6289	1.76	6.96
MLE	0.1201	12.61	13.29	0.1201	1.41	5.06
Max-Utility	0.1299	12.17	14.06	0.1299	1.79	5.57
GAN-Utility	0.1304	12.01	**14.35**	0.1304	2.69	6.12
Specificity-MLE	0.1023	12.61	13.53	**0.1640**	**4.45**	**7.85**
Specificity-GAN-Utility	**0.1012**	**12.84**	14.18	0.1357	2.95	6.08

Table 6.2: Diversity as measured by the proportion of unique trigrams in model outputs. Bleu and Meteor scores are calculated using an average of 6 references under *generic* setting and using an average of 3 references under *specific* setting. The highest numbers within a column is in bold (except for diversity under *generic* setting where the lowest number is bold).

Table 6.1 shows the result of our specificity classifier. We evaluate using 10-fold cross validation on our labelled set of 3310 questions. We perform feature ablation where we evaluate the performance of our model using each of the feature sets separately. Similar to Louis and Nenkova (2011), we find that syntax and polarity are strong indicators of specificity whereas question length is comparatively a weak indicator, even though intuitively we might think length to be a strong indicator since specific questions tend to be longer. Under specificity features, we find that path in WordNet feature to be more useful than the Inverse Document Frequency feature. Similar to Louis and Nenkova (2011), we find that the question bag-of-words feature to be

the most useful. Among the newly introduced features, we find the average word embeddings feature is more useful that the features that calculate the similarity of the question to the context.

Our best model is the one that uses all the features and attains an accuracy of 0.73 on the test set. In comparison, a baseline model that predicts the specificity label at random gets an accuracy of 0.58 on the test set.

6.6.2 Question Generation Results

Table 6.2 compares the performance of our specificity-controlled question generation model to the question generation models described in the previous chapter. We aim to evaluate how good are these models at generating questions at a given level of specificity. In our amazon dataset, each context is paired with upto 10 reference questions. We use our specificity classifier to identify *generic* reference questions and *specific* reference questions. We then use our evaluation metrics BLEU and METEOR to compare the model outputs to *generic* references and *specific* references separately. We call these *generic* and *specific* settings respectively. In case of the Lucene, MLE, Max-Utility and GAN-Utility models, the same model output is compared to the references in the two cases. Whereas in case of Specificity-MLE and Specificity-GAN-Utility models, under generic setting, the *generic* references are compared to the model output when the context is append with the "<generic>" token, whereas under specific setting, the *specific* references are compared to the model output when the context is append with the "<specific>" token. DIVERSITY

is measured using the proportion of unique trigrams in the model output.

Under *generic* setting, we find that given a context appended with a "<generic>" token, the specificity-controlled models (Specificity-MLE & Specificity-GAN-Utility) generates questions that is at a lower DIVERSITY than the other models. Whereas, under *specific* setting, we find that given a context appended with a "<specific>" token, these models generate questions with a higher DIVERSITY compared to the other models. This shows that our specificity-controlled models are capable of generating questions are varied diversity, thus varied specificity.

Under *specific* setting, we find that the Specificity-MLE model generates questions that get much higher BLEU and METEOR scores when compare to the *specific* reference questions compared to the other models. Under *generic* setting, however, we find that the specificity-controlled models generate questions that are at a similar BLEU and METEOR scores as the other models. This suggests that the specificity-controlled models tend to be more closer to the *specific* reference questions than to the *generic* reference questions. Interestingly, unlike the results from the previous chapter, a maximum-likelihood (MLE) training objective seems to be more effective for training a specificity-controlled question generation model than the more sophisticated GAN-Utility training objective.

Table 6.3 shows two example product descriptions and the questions generated by different models. As you can see, the specificity-controlled models generate more specific and more generic questions compared to other models.

6.7 Conclusion

In this chapter, we described our specificity-controlled question model which given a context and a level of specificity, generates a question at that desired level of specificity. We train a specificity classifier which given a context and a question can predict the level of specificity of the question to the context with 73% accuracy. We use this specificity classifier to automatically label all the questions in the training data of the question generation model described in the previous chapter. Further, we use the specificity label as additional signal during the training of the question generation model described in the previous chapter. We use automatic metric based evaluation to show that our specificity-controlled question generation model can generate questions that are more generic or more specific to the given context depending on the given input specificity level in comparison to other models.

Title	Signature sleep renewfoam infused memory foam and independently encased coil mattress , 8-inch
Product Description	Undecided between a coil mattress and a memory foam mattress ? Why not experience the best of both worlds with the signature sleep 8x201d; renewfoam coil mattress. The gel infused memory foam and coolmax; outer cover are perfectly paired to provide a fresh and cool sleeping surface, while the independently encased coils eliminate motion disturbance. With the signature sleep renewfoam coil mattress, always wake up feeling refreshed, rejuvenated and renewed.
Reference	do you need a separate box springs to go with this mattress ?
Lucene	how long does this matress last ?
MLE	what is the weight limit for this mattress ?
Max-Utility	what is the weight limit for this mattress ?
GAN-Utility	what are the dimensions of the mattress pad pad ?
Spec-MLE (g)	does it come with a cover ?
Spec-MLE (s)	does this mattress come with a box spring ?
Spec-GAN-Utility (g)	what is the warranty on this mattress ?
Spec-GAN-Utility (s)	what is the density of the mattress ?
Title	new cutting blade knife for kitchenaid mixer meat grinder; fga food chopper
Product Description	New sharp design cutting blade for the white fga kitchenaid meat grinder & food chopper. This knife is much improved from the original style cutter that came with the grinder attachment. You will see the improved difference when using a true cutting blade when grinding meat or vegetables. Stainless steel part with lifetime no rust guarantee from butcher-baker. Making sausage with our kitchenaid meat grinder ? We have the stainless steel stuffer tubes also. Need replacment meat grinder discs? We have them also. Add these parts to your order now for combined shipping discounts.
Reference	does this fit an older model kitchenaid mixer-grinder attachment fga model or not ? some reviewers are saying it does not fit ?
Lucene	can anyone confirm the dimensions of the square hole ?
MLE	will this fit the ?
Max-Utility	can this be used to grind almonds ?
GAN-Utility	does this blade fit the?
Spec-MLE (g)	does it come with a blade ?
Spec-MLE (s)	does this blade work with the kitchenaid professional model ?
Spec-GAN-Utility (g)	will this blade work with the weston model ?
Spec-GAN-Utility (s)	does this work well for a full size ? like a fine blade ?

Table 6.3: Example outputs from each of the systems for a single product description. g indicates generic token whereas s indicates specific token.

Chapter 7: Conclusion

7.1 Summary

In this dissertation we identify the importance of teaching machines to ask clarification questions i.e. questions that point at missing information in a given text. We propose to take a machine learning approach to clarification question generation where a model is trained using large amounts of (context, question) pairs. In order to do this learning effectively, we create datasets for two scenarios: technical support (StackExchange) and e-retail (Amazon). We present two approaches to the problem of clarification question generation. In the first approach, we develop a model which given a context, extracts a set of potential candidate questions from a pool of existing questions and then ranks them in the order of their usefulness to the given context. We model the usefulness of a question using the idea of expected value of perfect information: a good question is one whose expected answer will be useful. We find that "answer" helps in identifying good clarification questions. In the second approach, we develop a model which given a context, generates questions from scratch instead of ranking existing question. We train a sequence-to-sequence neural network model using the recent idea of Generative Adversarial Network (GAN) to maximize an answer-based reward function. We show that our adversarially trained

model generates questions that are more specific to the given context. We further explore the notion of controlling the specificity of generated question by explicitly training a question generation model which given a context and a level of specificity (generic or specific), generates a question at that level of specificity. To label the large number of questions in our training data with the level of specificity, we train a binary classifier which given a context and a question, predicts whether the question is specific (to the context) or generic. We include the level of specificity as an additional signal during the training of our question generation model and find that our specificity-controlled question generation model can generate questions at a desired level of specificity.

7.2 Future Directions

In this section, we discuss some potential future directions of research in the area of clarification question generation.

7.2.1 Using Multi-modal Context

The question generation models proposed in this work only make use of textual context. However, often contexts include other modals of information as well. For instance, textual descriptions of products on amazon.com are paired with the image of the product. We can make use of the image to ask more relevant questions. Consider the description of a cookware set in Figure 7.1. A question generation model that uses only the textual context might generate the question "Does the

Figure 7.1: An example of product description on amazon.com paired with the image of the product.

set include a ladle?" since the description does not contain the details of the items included in the cookware set. However, if the model were to use the image of the product as well, then it could find that the ladle is already included and hence would not generate such a redundant question. Thus, a potential future direction would be to use both textual and image contexts to train a question generation model.

7.2.2 Using External Knowledge Sources

The models described in this work learn to ask a clarification question by looking at previously asked questions in a similar context. More specifically, we rely on our data to include the kinds of questions that we would like to ask. The main purpose of generating the clarification questions is to identify the missing information in a given text. To understand what is missing, one needs to first know what should have been there. As humans, we rely on our prior knowledge

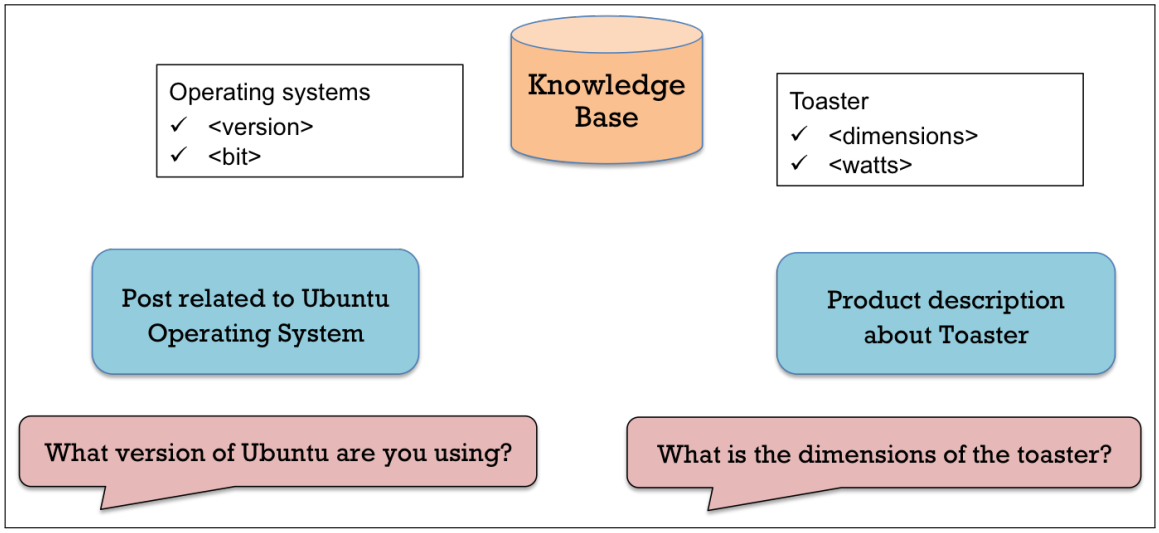

Figure 7.2: Example of a question generation model that uses a knowledge base containing attributes of an operating system (or attributes of a toaster) to ask a relevant clarification question.

about the subject to decide what should have been there but is missing and then ask a clarification question pointing out the missing information. Therefore, one potential extension to our work would be to automatically extract information from existing knowledge sources and makes use of it to generate a clarification question. For instance, in Figure 7.2, given a post related to Ubuntu operating system, if the model had access to a knowledge base that contained the information that operating systems differ by versions and bits, then the model could use that information to generate a question. Similarly, in the context of Amazon, if the model had access to a knowledge base containing various attributes of a product, then it could use that to understand what information is missing from the given description and ask a useful question.

7.2.3 Interactive Search Queries

With the emergence of internet, vast amounts of data is stored online. We frequently use search engines to extract relevant information from this abundance of online data. However, we might often find ourselves sifting through the search results when our original search query is not specific enough. In such a scenario, it might have been useful if the search engine would have asked us a follow-up question. For instance, if a user query is "How long does it take to get a PhD?", the search engine could ask the user "In which field?" because the answer would differ based on the field of study. Likewise, if a user query is "Historical gas prices", the search engine could ask "Which region?" or "Which year?" because the prices would differ by region and year. Thus, a potential future direction of our work would be to train a clarification question generation model which given a search query can generate follow-up question(s) that can help narrow down the original query.

7.2.4 Question Asking in Writing Assistance

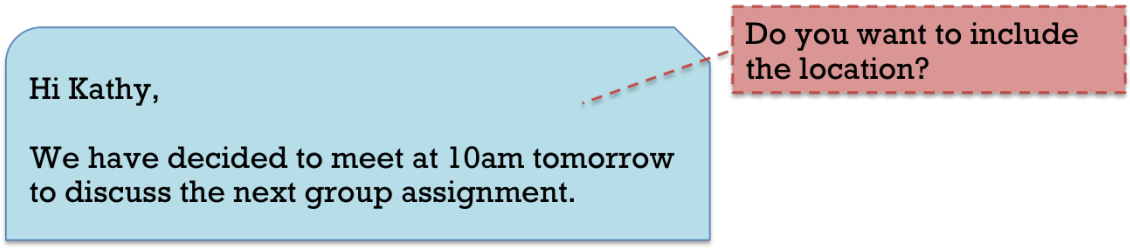

Figure 7.3: An example of a writing assistance tool which given a content, identifies the missing information and asks a question about it.

In our day-to-day lives, we frequently use computers for writing documents, emails, etc. With the advancements of technologies, many of the text processing tools these days help us write better by pointing our spelling errors or other minor grammatical errors. However, these tools still are not at par with humans when it comes to suggesting content level changes. For example, consider a scenario where a student is writing their statement of purpose. A human reviewing this document might suggest changes such as possible addition of description of a project, addition of a missing reference to a related work, etc. Given the vast amounts of available data online, writing assistance tools might soon be able to suggest such informational changes. A first step towards this direction might be an email assistance tools that can point out missing information in your email. For instance, consider you have drafted an email such as the one in Figure 7.4. Since you have forgotten to mention the location of the meeting, Kathy might send you a follow-up email asking for the location. Such a follow-up email exchange could have been avoided if the email application could have suggested to include the location in the first place.

7.2.5 Towards Intelligent Dialogue Agents

Asking questions is one of the key components of a conversation. Humans often ask questions to miss information gaps during a conversation. Therefore, in order for automated agents to be successful at conversations, it is important that we teach these agents to ask intelligent questions. Consider a scenario where I have asked a robot to get me my coffee mug from the kitchen (Figure 7.4). If there are multiple

Figure 7.4: An example conversation with a robot where the robot asks questions to resolve its uncertainty.

mugs in the kitchen, an intelligent robot would ask a question such as "What color is your coffee mug?" to resolve this ambiguity. Further, if I reply by saying that the color of my mug is black, and if the robot finds multiple black mugs in the kitchen, it could ask a follow-up question to further resolve the ambiguity. Teaching robots to ask such useful questions would enable them to be more intelligent.

7.2.6 Question Asking to Help Build Reasoning

Asking intelligent questions can also be used as a tool for enabling automated building of reasoning. For example, consider a robot is reading the passage shown in Figure 7.5. As it is reading this passage, assume it is building an understanding of the world. Suppose the robot asks a question such as "Why was Jill upset?" as it is building this reasoning. And a human answers the robot by saying "Because she did not win the race.". This will help the robot understand that reaching the

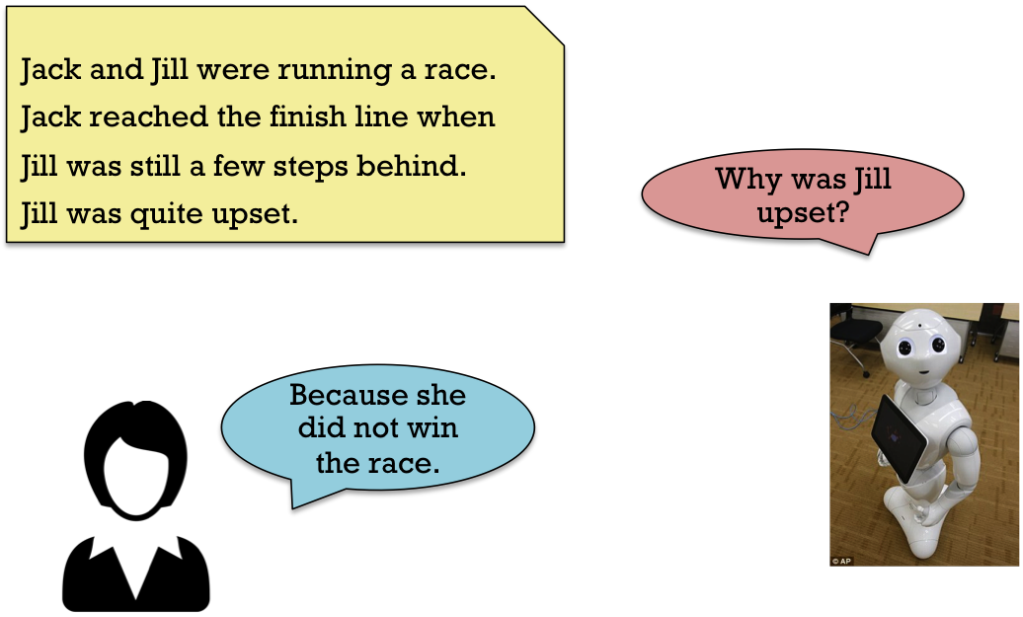

Figure 7.5: An example scenario where a robot is reading a passage and asking questions to a human to build an understanding of the world.

finish line leads to winning the race and not winning a race would make someone upset. The robot could then go ahead and update its understanding of the world using these reasonings.

7.2.7 Generalization Beyond Large Datasets

In this dissertation, we have described methods for generating clarification questions that rely heavily on learning from large datasets. In future, we would want to be able to generate questions without going through the same substantial dataset-building process. One method for this would be to bootstrap the process by using template based approach (or humans) to initially generate some small set of questions. Then train our model on this small set to generate more questions.

And finally use these generated questions to further retrain our model. Second method of generalization would be using the idea of domain adaptation where we could use large amounts of existing out-of-domain data to train a model and then use small amounts of in-domain data to tune the model. Lastly, we could modify existing reading comprehension datasets to create clarification questions dataset by removing the answer sentence from the passage and then using the question as the clarification question and the passage as the context.

Appendix A: Crowdsourcing Annotation Details

A.1 Question Ranking Task Evaluation

In this section we describe the details of the process of collecting human judgments for the evaluation of the outputs of our question ranking model described in Chapter 4. We use Upwork[1] for collecting our expert human judgments. Upwork is a platform which allows us to post a job description and recruit people specifically for a task.

We show the following instructions to the annotator:

Your task is to ask the right question.

You will be shown a post to StackExchange that is incomplete: that is, in order to provide a useful solution to this post, the original poster needs to provide some additional information.

In order to elicit that additional information from the original poster, you want to ask a question.

You will be provided a list of ten possible questions that you can ask. You must provide two pieces of information:

1) Which of these questions is the single best one? If you could only ask one question,

[1] https://upwork.com

which one would you ask?

2) Which other questions would be valid to ask, even if not best.

The interface will force you to choose a single best question by marking it with a radio button, and other valid questions with check boxes.

Some of these are hard. Try your best to answer them. It took us 5-6 minutes per example, so please don't rush.

After every question you'll be asked for your confidence in your selection of the 'best' question. For some of them you may just have to take an educated guess, for others you will be quite sure.

Note: 'Best' by definition is also 'valid': so whatever you select as 'best' you should also mark as 'valid'.

We show the Upwork annotators the following interface for performing the task of annotating the one "best" question and one or more "valid" questions, given a post from StackExchange dataset.

A.2 Question Generation Task Evaluation

In this section we describe the details of human based evaluation process for evaluating outputs of the question generation models described in Chapter 5.

Figure A.2 is overview of the task shown to the annotators.

Figure A.3 is the set of instructions shown to the annotators.

Figure A.4 is the set of rules and tips shown to the annotators.

Figure A.5 shows two example annotations shown to the annotators.

Figure A.6 shows the interface shown to the annotators.

A.3 Specificity Labeling Task

In this section we describe the details of the annotation task for labeling questions with their specificity levels presented in Chapter 6. Figure A.7 shows the instructions shown to the annotators.

Figure A.8 shows the rules and tips shown to the annotators.

Figure A.9 shows an example annotation shown to the annotators to guide them to do the task.

Figure A.10 shows the interface shown to the annotators.

Title: No wifi after restart in Ubuntu 16.04

After upgrading to 16.04, there is no wifi whenever I restart the system. My Wireless interface of Ubuntu is RT3290 Wireless 802.11n 1T/1R PCIe

on iwconfig I got the following

lo no wireless extensions.

wlan0 IEEE 802.11bgn ESSID:off/any
Mode:Managed Access Point: Not-Associated Tx-Power=0 dBm
Retry short limit:7 RTS thr:off Fragment thr:off
Power Management:off

eth0 no wireless extensions.

Currently to start wifi again I have to shutdown, then boot the system again.

How to fix the problem?

Best	Valid	Question
○	☐	if i post it as an answer , would you kindle mark as such ?
○	☐	what is 4g wifi connection ?
○	☐	what exactly do you mean by make fails ?
○	☐	i doubt it , shutdown and reboot are exactly identical ! are you really rebooting ?
○	☐	be clear about the problem . is ubuntu not showing them even though they are present ?
○	☐	is ubuntu detecting yuor wireless card ? **iwconfig** does list your card ?
○	☐	which ubuntu 15 ?
○	☐	can you type `iwconfig` in terminal and paste what it returns here ?
○	☐	welcome to ask ubuntu ! ; -) is the wireless lan disabled in the bios ?
○	☐	what does this tell us ?

Confidence in "Best":
○ Educated guess (Low)
○ Pretty sure (Medium)
○ Quite sure (High)

Figure A.1: Example of the interface shown to annotators on UpWork for annotating "best" and "valid" questions, given a post.

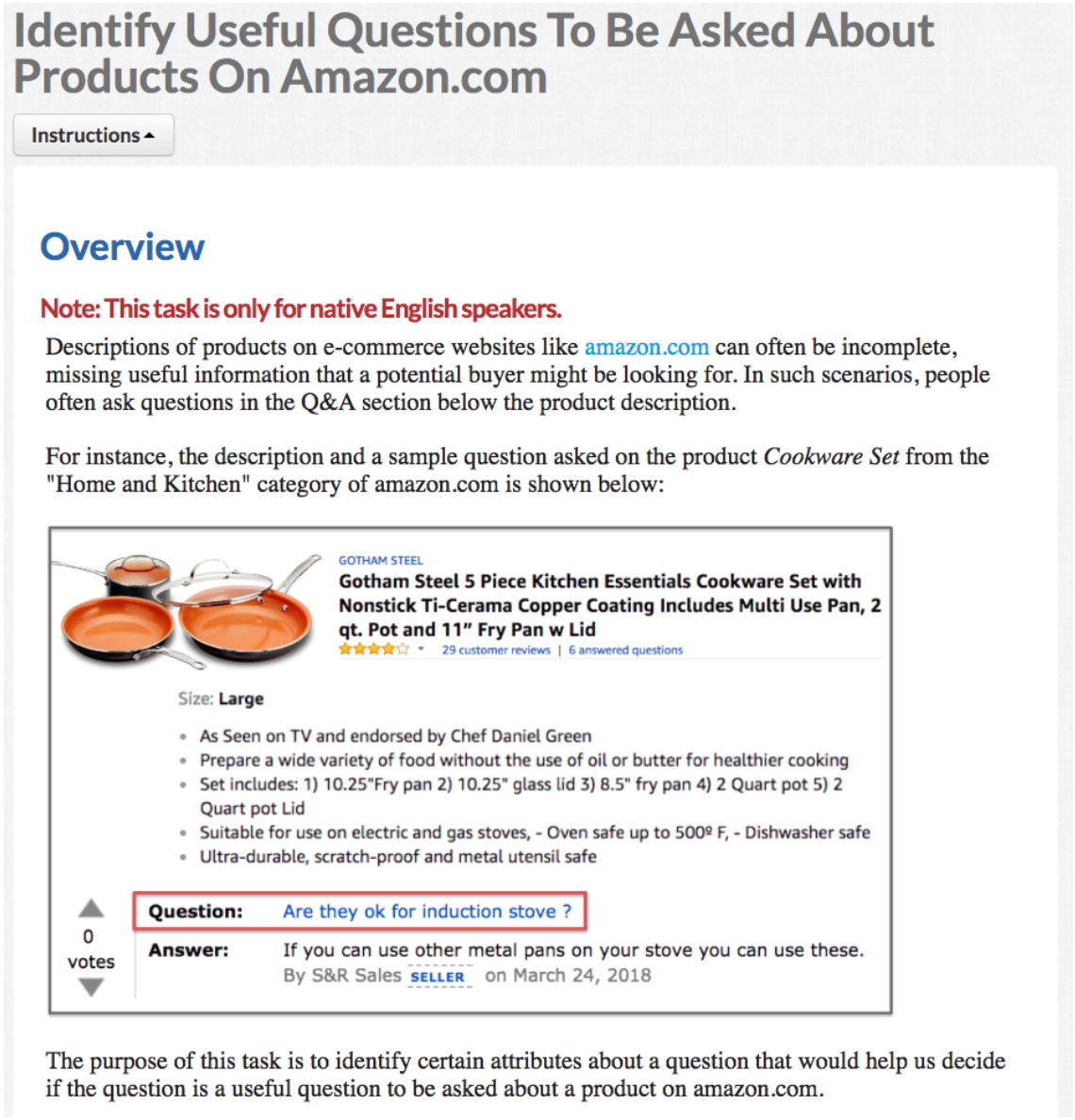

Figure A.2: Task overview shown to annotators on Figure-Eight for the task of evaluating model generated questions.

Instructions

You will be shown the description of a product from the "Home and Kitchen" category from Amazon. com along with a question that one could ask about the product. Note that the description and the question will be displayed in a format different from the one shown above (See examples at the end).

Your task is to answer the following about the question:

a. Is the question on topic? (Select among: Yes or No)

- A question is on topic if it is relevant to the product.
- A useful question will be on topic.

b. Is the question grammatical?

You will be asked to chose from the following options:

- Grammatical: Question is grammatically well-formed (Note: Ignore case since all words are lowercased)
- Comprehensible: Question may have some grammatical errors or typos but is understandable.
- Incomprehensible: Question does not make sense.

c. How specific is the question?

You will be asked to chose from the following options:

- Specific pretty much only to this product
- Specific to this and other very similar products (or the same product from a different manufacturer)
- Generic enough to be applicable to many other products of this type
  - E.g. "is this dishwasher safe?", "what is the height of the table?"
- Generic enough to be applicable to any product under Home and Kitchen
  - E.g. "what is the warranty?", "is it made in usa?", "what are the dimensions?"
- N/A (Not applicable): Question is not on topic OR is incomprehensible

d. Does the question ask for new information currently not included in the description?

You will be asked to chose from the following options:

- Completely: Question is asking for information that is completely missing from the description
- Somewhat: Question is asking for information that is partly included in the description
- No: Question is asking for information that is included in the description
- N/A (Not applicable): Question is not on topic OR is incomprehensible

e. How useful is the question to a potential buyer (or a current user) of the product?

You will be asked to chose from the following options:

- Useful enough to be included in the product description
- Useful to a large number of potential buyers (or current users)
- Useful to a small number of potential buyers (or current users)
- Useful only to the person asking the question
- N/A (Not applicable): Question is not on topic OR is incomprehensible OR is not asking for new information

Figure A.3: Instructions shown to annotators on Figure-Eight for the task of evaluating model generated questions.

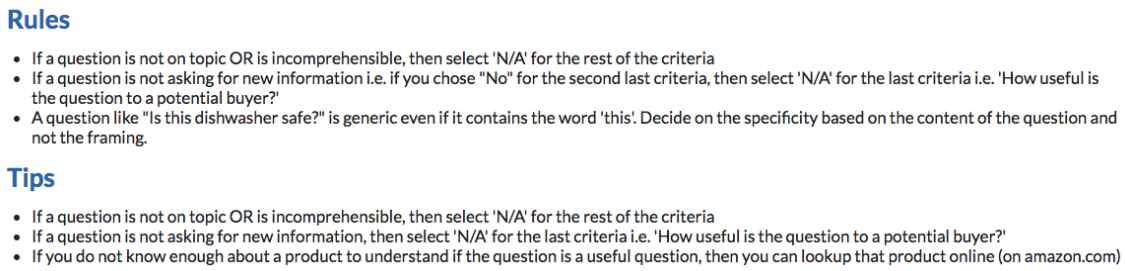

Figure A.4: Rules and tips shown to annotators on Figure-Eight for the task of evaluating model generated questions.

Example 1

Product description

Sofab Lass Love Seat, Chocolate Sofab is a revolutionary seating product from Albany Industries. It combines great looks, comfortable seating and durability into sofas, love seats, chairs and ottomans that can be shipped UPS Ground. Some assembly required.

Question: "does it come with the pillows shown?"

Answer the following regarding the question above:

- Is the question on topic?
  - Yes
- Is the question grammatical?
  - Grammatical
- How specific is the question?
  - **Specific pretty much only to this product**
- Does the question ask for new information currently not included in the description?
  - Completely
- How useful is the question to a potential buyer (or a current user) of the product?
  - **Useful to a large number of potential buyers (or current users)**

Example 3

Product description

Wolf Twin Bunkie Board for Bunk Beds The Wolf Bunkie Board is made from natural wood and wrapped in a grey cloth. It is intended to provide a foundation for a bunk bed for additional support. Overall Dimensions: 2" H x 38" W x 75" D Weight: 25 lbs

Question: "what is the weight limit for this ?"

Answer the following regarding the question above:

- Is the question on topic?
  - Yes
- Is the question grammatical?
  - Grammatical
- How specific is the question?
  - **Generic enough to be applicable to many other products of this type**
- Does the question ask for new information currently not included in the description?
  - Completely
- How useful is the question to a potential buyer (or a current user) of the product?
  - Useful enough to be included in the product description

Figure A.5: Example annotations shown to annotators on Figure-Eight for the task of evaluating model generated questions.

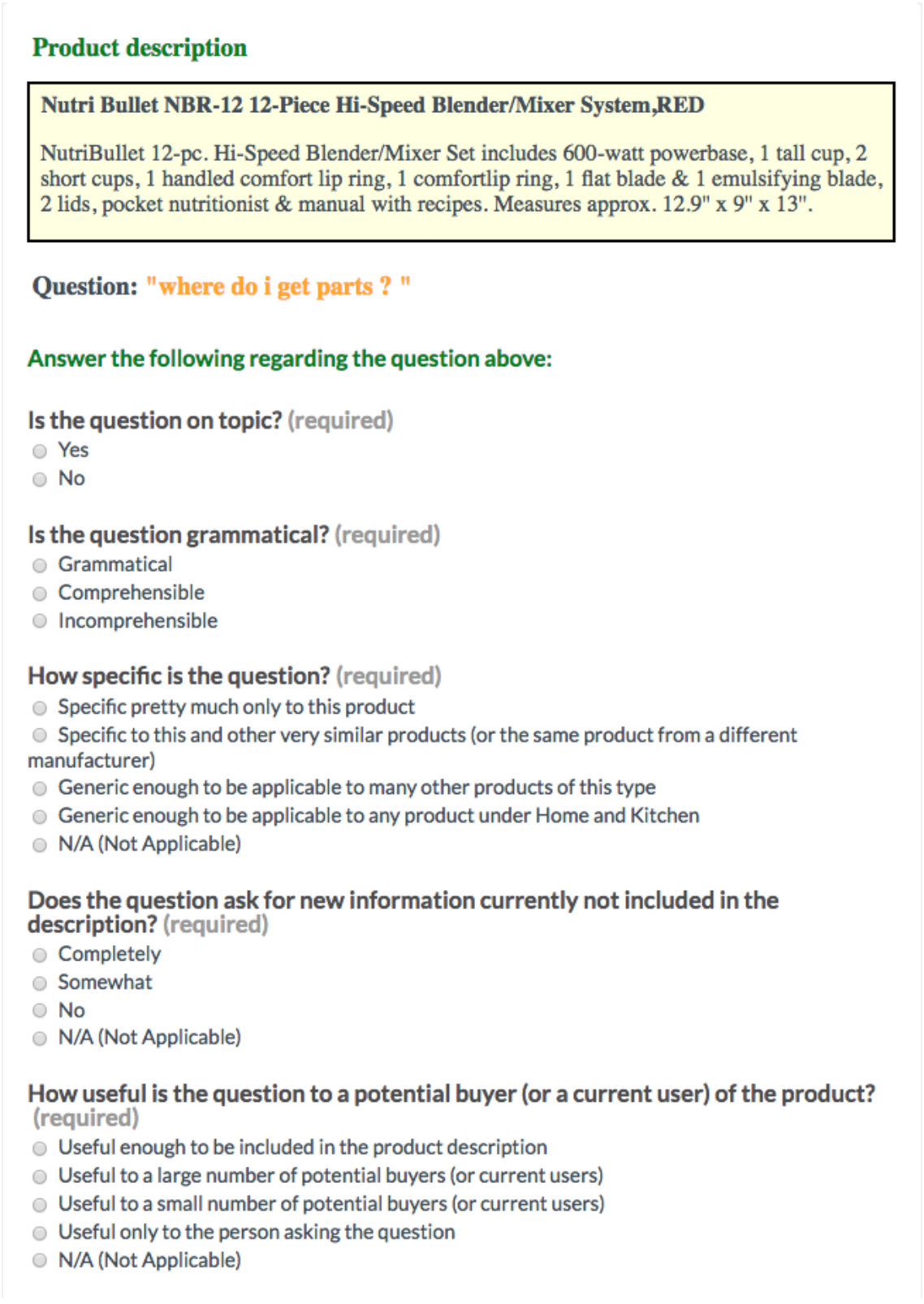

Figure A.6: Interface shown to the annotators on Figure-Eight for the task of evaluating model generated questions.

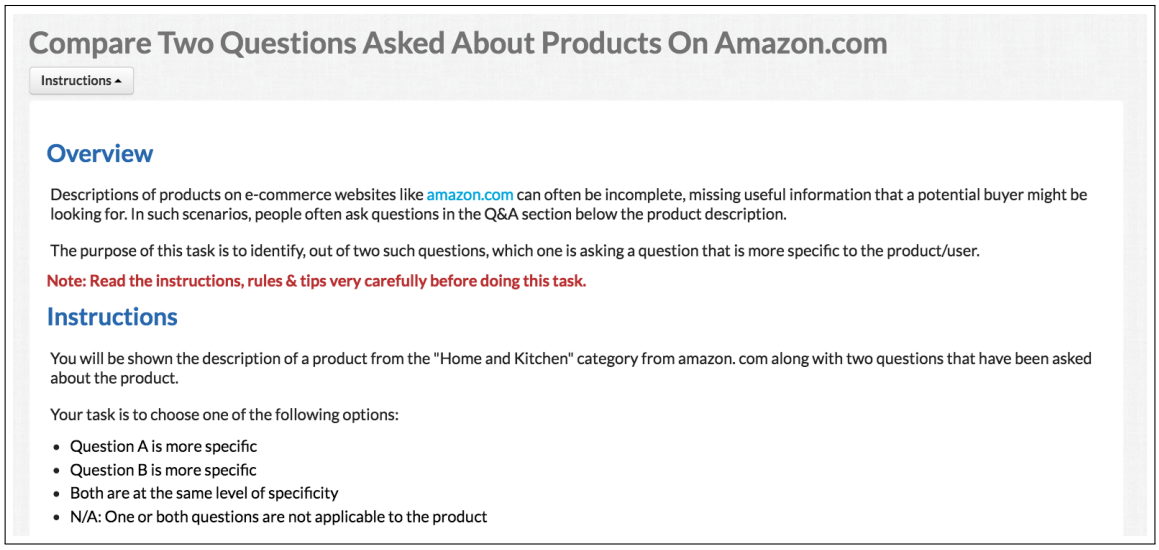

Figure A.7: Instructions shown to the annotators for the task of comparing the specificity of two questions asked about a product on amazon.com .

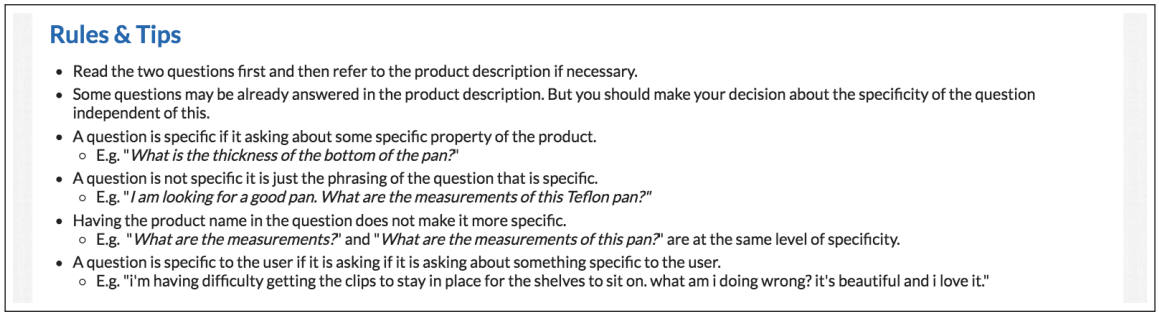

Figure A.8: Rules and Tips shown to the annotators for the task of comparing the specificity of two questions asked about a product on amazon.com .

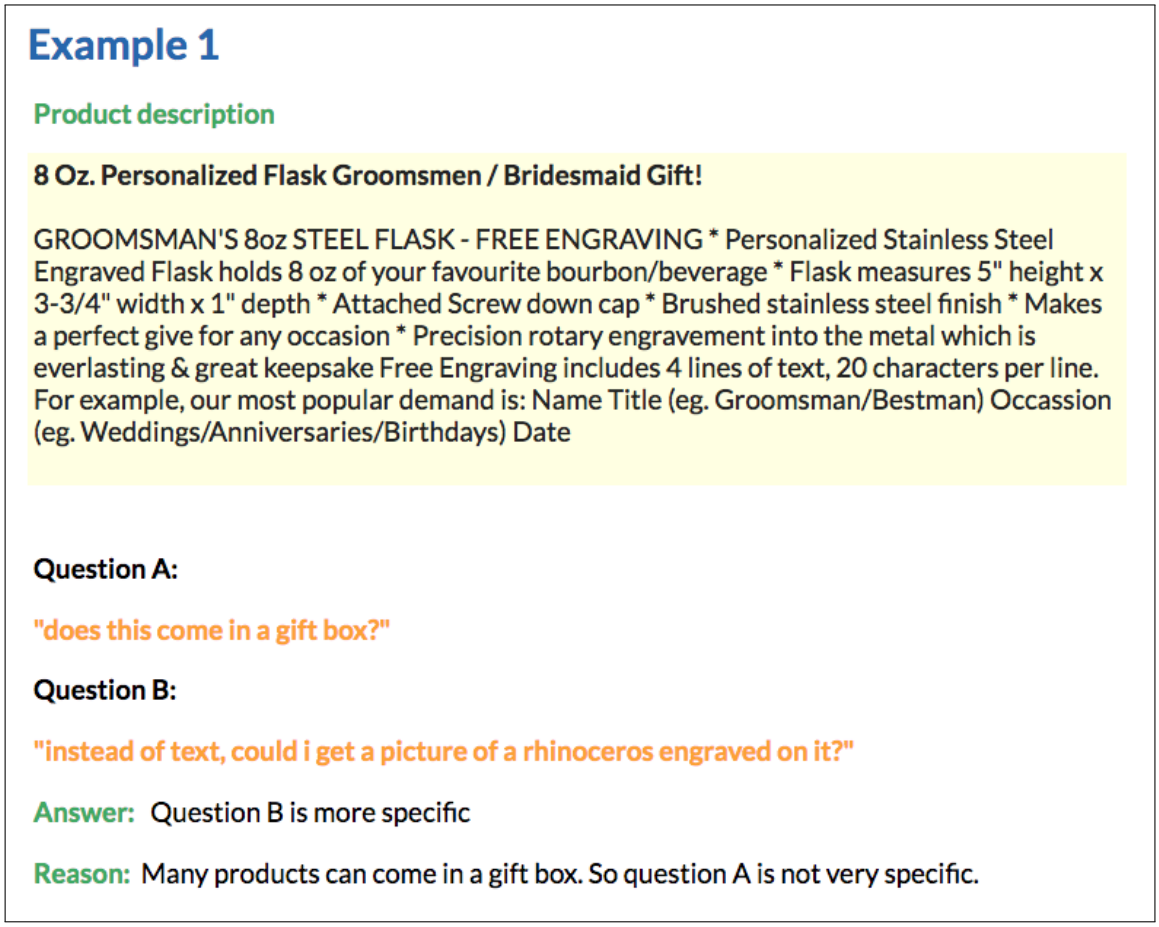

Figure A.9: Example shown to the annotators for the task of comparing the specificity of two questions asked about a product on amazon.com .

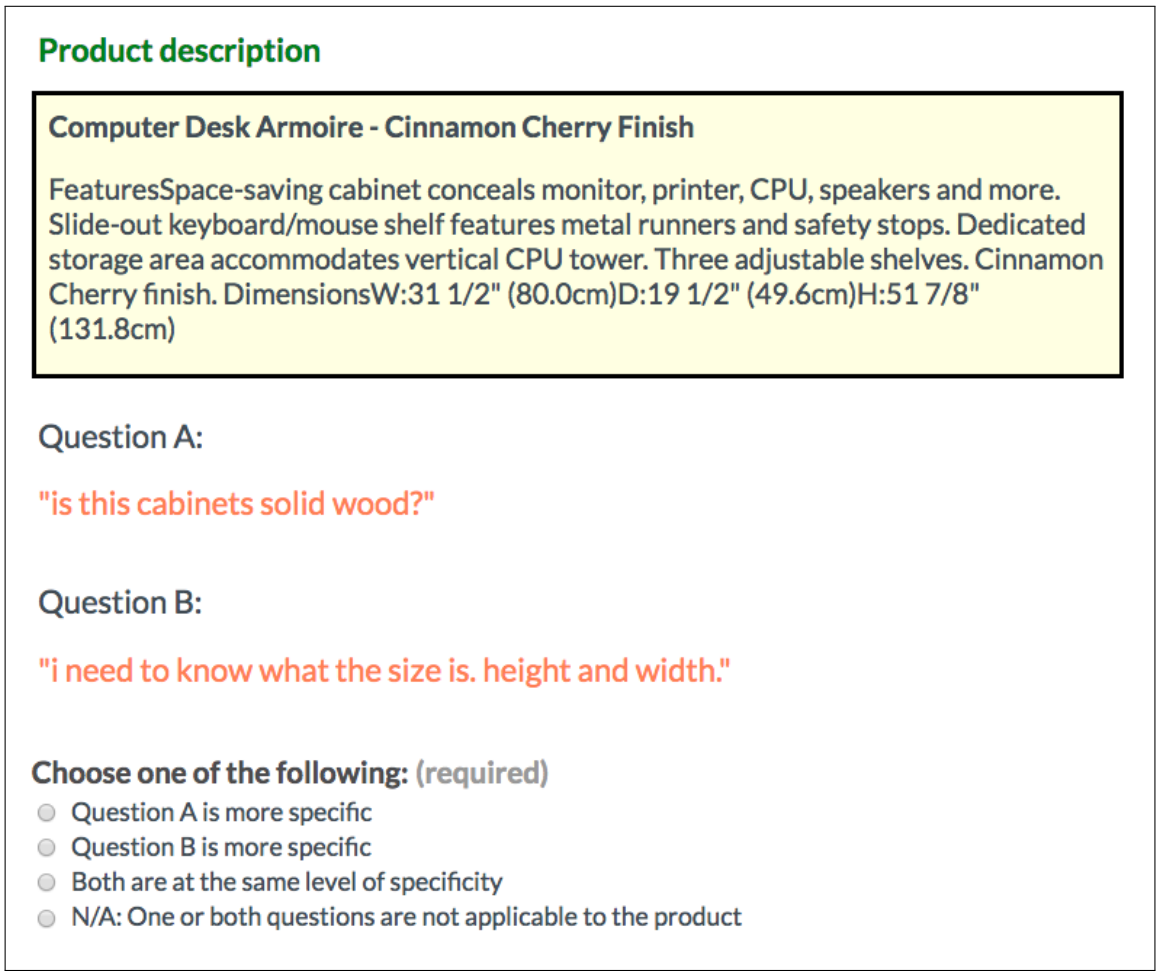

Figure A.10: Interface shown to the annotators for the task of comparing the specificity of two questions asked about a product on amazon.com .

Bibliography

Husam Ali, Yllias Chali, and Sadid A Hasan. 2010. Automation of question generation from sentences. In *Proceedings of QG2010: The Third Workshop on Question Generation*. pages 58–67.

Jens Allwood. 2000. An activity based approach to pragmatics. *Abduction, belief and context in dialogue: Studies in computational pragmatics* pages 47–80.

Jacob Andreas and Dan Klein. 2016. Reasoning about pragmatics with neural listeners and speakers. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 1173–1182.

Yoav Artzi and Luke Zettlemoyer. 2011. Bootstrapping semantic parsers from conversations. In *Proceedings of the conference on empirical methods in natural language processing*. Association for Computational Linguistics, pages 421–432.

Muhammad Asaduzzaman, Ahmed Shah Mashiyat, Chanchal K Roy, and Kevin A Schneider. 2013. Answering questions about unanswered questions of stack overflow. In *Proceedings of the 10th Working Conference on Mining Software Repositories*. IEEE Press, pages 97–100.

Mordecai Avriel and AC Williams. 1970. The value of information and stochastic programming. *Operations Research* 18(5):947–954.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*. pages 65–72.

Daniel G Bobrow, Ronald M Kaplan, Martin Kay, Donald A Norman, Henry Thompson, and Terry Winograd. 1977. Gus, a frame-driven dialog system. *Artificial intelligence* 8(2):155–173.

Antoine Bordes and Jason Weston. 2016. Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683* .

Julian Brooke and Graeme Hirst. 2014. Supervised ranking of co-occurrence profiles for acquisition of continuous lexical attributes. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. pages 2172–2183.

Julian Brooke, Tong Wang, and Graeme Hirst. 2010. Automatic acquisition of lexical formality. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*. Association for Computational Linguistics, pages 90–98.

Christian Buck, Jannis Bulian, Massimiliano Ciaramita, Andrea Gesmundo, Neil Houlsby, Wojciech Gajewski, and Wei Wang. 2017. Ask the right questions: Active question reformulation with reinforcement learning. *arXiv preprint arXiv:1705.07830* .

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. pages 740–750.

Wei Chen. 2009. Aist, g., mostow, j.: Generating questions automatically from informational text. In *Proceedings of the 2nd Workshop on Question Generation (AIED 2009)*. pages 17–24.

Chenhui Chu, Raj Dabre, and Sadao Kurohashi. 2017. An empirical comparison of domain adaptation methods for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. volume 2, pages 385–391.

Herbert H Clark. 1981. Definite reference and mutual knowledge. *Elements of discourse understanding* pages 10–63.

Herbert H Clark. 1996. Using language. 1996. *Cambridge University Press: Cambridge* 952:274–296.

Herbert H Clark, Susan E Brennan, et al. 1991. Grounding in communication. *Perspectives on socially shared cognition* 13(1991):127–149.

Herbert H Clark and Thomas B Carlson. 1982. Hearers and speech acts. *Language* pages 332–373.

Anni Coden, Daniel Gruhl, Neal Lewis, and Pablo N Mendes. 2015. Did you mean a or b? supporting clarification dialog for entity disambiguation. In *SumPre-HSWI@ ESWC*.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 1342–1352.

Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. 2017. Question generation for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 866–874.

Philip Edmonds and Graeme Hirst. 2002. Near-synonymy and lexical choice. *Computational linguistics* 28(2):105–144.

Ute Essen and Volker Steinbiss. 1992. Cooccurrence smoothing for stochastic language modeling. In *icassp*. IEEE, pages 161–164.

Song Feng, Ritwik Banerjee, and Yejin Choi. 2012. Syntactic stylometry for deception detection. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*. Association for Computational Linguistics, pages 171–175.

Jessica Ficler and Yoav Goldberg. 2017. Controlling linguistic style aspects in neural language generation. *Proceedings of the Workshop on Stylistic Variation, Empirical Methods in Natural Language Processing 2017* .

Alejandro Figueroa and Günter Neumann. 2013. Learning to rank effective paraphrases from query logs for community question answering. In *Association for the Advancement of Artificial Intelligence*. volume 13, pages 1099–1105.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th international conference on machine learning (ICML-11)*. pages 513–520.

David Goddeau, Helen Meng, Joseph Polifroni, Stephanie Seneff, and Senis Busayapongchai. 1996. A form-based dialogue manager for spoken language applications. In *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on*. IEEE, volume 2, pages 701–704.

Dave Golland, Percy Liang, and Dan Klein. 2010. A game-theoretic approach to generating spatial descriptions. In *Proceedings of the 2010 conference on empirical methods in natural language processing*. Association for Computational Linguistics, pages 410–419.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. 2016. *Deep learning*, volume 1. MIT press Cambridge.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems*. pages 2672–2680.

Art Graesser, Vasile Rus, and Zhiqiang Cai. 2008. Question classification schemes. In *Proc. of the Workshop on Question Generation*.

Arthur C Graesser, Natalie Person, and John Huber. 1992. Mechanisms that generate questions. *Questions and information systems* pages 167–187.

Arthur C Graesser and Natalie K Person. 1994. Question asking during tutoring. *American educational research journal* 31(1):104–137.

H Paul Grice. 1975. Logic and conversation. *1975* pages 41–58.

Barbara J Grosz and Candace L Sidner. 1986. Attention, intentions, and the structure of discourse. *Computational linguistics* 12(3):175–204.

Michael Heilman. 2011. *Automatic factual question generation from text*. Ph.D. thesis, Carnegie Mellon University.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

John J Hopfield. 1982. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences* 79(8):2554–2558.

Eduard Hovy. 1987. Generating natural language under pragmatic constraints. *Journal of Pragmatics* 11(6):689–719.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *International Conference on Machine Learning*. pages 1587–1596.

Diana Inkpen and Graeme Hirst. 2006. Building and using a lexical knowledge base of near-synonym differences. *Computational linguistics* 32(2):223–262.

Harsh Jhamtani, Varun Gangal, Eduard Hovy, and Eric Nyberg. 2017. Shakespearizing modern language using copy-enriched sequence to sequence models. In *Proceedings of the Workshop on Stylistic Variation*. pages 10–19.

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 of Computational Linguistics* 5(1):339–351.

Jad Kabbara and Jackie Chi Kit Cheung. 2016. Stylistic transfer in natural language generation systems using recurrent neural networks. In *Proceedings of the Workshop on Uphill Battles in Language Processing: Scaling Early Achievements to Robust Methods*. pages 43–47.

Tomoyuki Kajiwara and Mamoru Komachi. 2016. Building a monolingual parallel corpus for text simplification using sentence similarity based on alignment between word embeddings. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. pages 1147–1158.

Saidalavi Kalady, Ajeesh Elikkottil, and Rajarshi Das. 2010. Natural language question generation using syntax and keywords. In *Proceedings of QG2010: The Third Workshop on Question Generation*. questiongeneration. org, volume 2.

Jaap Kamps, Maarten Marx, Robert J Mokken, Maarten De Rijke, et al. 2004. Using wordnet to measure semantic orientations of adjectives. In *Language Resources and Evaluation Conference*. Citeseer, volume 4, pages 1115–1118.

Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. 2018. Sharp nearby, fuzzy far away: How neural language models use context. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1.

Catherine KOBUS, Josep Crego, and Jean Senellart. 2017. Domain control for neural machine translation. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*. pages 372–378.

Moshe Koppel, Shlomo Argamon, and Anat Rachel Shimoni. 2002. Automatically categorizing written texts by author gender. *Literary and Linguistic Computing* 17(4):401–412.

Moshe Koppel, Jonathan Schler, and Shlomo Argamon. 2009. Computational methods in authorship attribution. *Journal of the American Society for information Science and Technology* 60(1):9–26.

Moshe Koppel, Jonathan Schler, and Shlomo Argamon. 2011. Authorship attribution in the wild. *Language Resources and Evaluation* 45(1):83–94.

Igor Labutov, Sumit Basu, and Lucy Vanderwende. 2015. Deep questions without deep understanding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. volume 1, pages 889–898.

Shibamouli Lahiri, Prasenjit Mitra, and Xiaofei Lu. 2011. Informality judgment at sentence level and experiments with formality score. In *International Conference on Intelligent Text Processing and Computational Linguistics*. Springer, pages 446–457.

Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. 1990. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*. pages 396–404.

Oliver Lemon, Kallirroi Georgila, James Henderson, and Matthew Stuttle. 2006. An isu dialogue system exhibiting reinforcement learning of dialogue policies: generic slot-filling in the talk in-car system. In *Proceedings of the Eleventh Conference of the European Chapter of the Association for Computational Linguistics: Posters & Demonstrations*. Association for Computational Linguistics, pages 119–122.

Will Lewis, Christian Federmann, and Ying Xin. 2015. Applying cross-entropy difference for selecting parallel training data from publicly available sources for conversational machine translation. In *International Workshop on Spoken Language Translation*.

Jiwei Li, Michel Galley, Chris Brockett, Georgios Spithourakis, Jianfeng Gao, and Bill Dolan. 2016a. A persona-based neural conversation model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 994–1003.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016b. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 1192–1202.

Yikang Li, Nan Duan, Bolei Zhou, Xiao Chu, Wanli Ouyang, Xiaogang Wang, and Ming Zhou. 2018. Visual question generation as dual task of visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 6116–6124.

Ming Liu, Rafael A Calvo, and Vasile Rus. 2010. Automatic question generation for literature review writing support. In *International Conference on Intelligent Tutoring Systems*. Springer, pages 45–54.

Annie Louis and Ani Nenkova. 2011. Automatic identification of general and specific sentences by leveraging discourse annotations. In *Proceedings of 5th International Joint Conference on Natural Language Processing*. pages 605–613.

Ryan Lowe, Nissan Pow, Iulian V Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Special Interest Group on Discourse and Dialogue*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Empirical Methods in Natural Language Processing*.

David M Markowitz and Jeffrey T Hancock. 2016. Linguistic obfuscation in fraudulent science. *Journal of Language and Social Psychology* 35(4):435–445.

Thomas A Mathew. 2009. *Supervised categorization of habitual versus episodic sentences*. Ph.D. thesis, Georgetown University.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, pages 43–52.

Julian McAuley and Alex Yang. 2016. Addressing complex and subjective product-related queries with customer reviews. In *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, pages 625–635.

Paul Michel and Graham Neubig. 2018. Extreme adaptation for personalized neural machine translation. In *Association for Computational Linguistics*.

Tomas Mikolov. 2010. Recurrent neural network based language model. In *Interspeech*.

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.

George A Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J Miller. 1990. Introduction to wordnet: An on-line lexical database. *International journal of lexicography* 3(4):235–244.

Hideki Mima, Osamu Furuse, and Hitoshi Iida. 1997. Improving performance of transfer-driven machine translation with extra-linguistic informatioon from context, situation and environment. In *International Joint Conferences on Artificial Intelligence (2)*. pages 983–989.

Alejandro Mosquera and Paloma Moreda. 2012. Smile: An informality classification tool for helping to assess quality and credibility in web 2.0 texts. In *Proceedings of the International Association for the Advancement of Artificial Intelligence Conference on Web and Social Media workshop: Real-Time Analysis and Mining of Social Streams (RAMSS)*.

Nasrin Mostafazadeh, Chris Brockett, Bill Dolan, Michel Galley, Jianfeng Gao, Georgios Spithourakis, and Lucy Vanderwende. 2017. Image-grounded conversations: Multimodal context for natural question and response generation. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. volume 1, pages 462–472.

Nasrin Mostafazadeh, Ishan Misra, Jacob Devlin, Margaret Mitchell, Xiaodong He, and Lucy Vanderwende. 2016. Generating natural questions about an image. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 1802–1813.

Preslav Nakov, Doris Hoogeveen, Lluís Màrquez, Alessandro Moschitti, Hamdy Mubarak, Timothy Baldwin, and Karin Verspoor. 2017. Semeval-2017 task 3: Community question answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. pages 27–48.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. 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.

Titas Nandi, Chris Biemann, Seid Muhie Yimam, Deepak Gupta, Sarah Kohail, Asif Ekbal, and Pushpak Bhattacharyya. 2017. Iit-uhh at semeval-2017 task 3: Exploring multiple features for community question answering and implicit dialogue identification. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. pages 90–97.

Masato Neishi, Jin Sakuma, Satoshi Tohda, Shonosuke Ishiwatari, Naoki Yoshinaga, and Masashi Toyoda. 2017. A bag of useful tricks for practical neural machine translation: Embedding layer initialization and large batch size. In *Proceedings of the 4th Workshop on Asian Translation (WAT2017)*. pages 99–109.

Xing Niu and Marine Carpuat. 2016. The UMD Machine Translation Systems at IWSLT 2016: English-to-French Translation of Speech Transcripts. In *International Workshop on Spoken Language Translation*.

Xing Niu, Marianna Martindale, and Marine Carpuat. 2017. A study of style in machine translation: Controlling the formality of machine translation output. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 2814–2819.

Xing Niu, Sudha Rao, and Marine Carpuat. 2018. Multi-task neural models for translating between styles within and across languages. In *Proceedings of the 27th International Conference on Computational Linguistics*. pages 1008–1021.

Andrew M Olney, Arthur C Graesser, and Natalie K Person. 2012. Question generation from concept maps. *Dialogue & Discourse* 3(2):75–99.

Naho Orita, Eliana Vornov, Naomi Feldman, and Hal Daumé III. 2015. Why discourse affects speakers' choice of referring expressions. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. volume 1, pages 1639–1649.

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*. Association for Computational Linguistics, pages 311–318.

Ellie Pavlick and Ani Nenkova. 2015. Inducing lexical style properties for paraphrase and genre differentiation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 218–224.

Ellie Pavlick and Joel Tetreault. 2016. An empirical analysis of formality in online communication. *Transactions of the Association for Computational Linguistics* 4:61–74.

James W Pennebaker, Martha E Francis, and Roger J Booth. 2001. Linguistic inquiry and word count: Liwc 2001. *Mahway: Lawrence Erlbaum Associates* 71(2001):2001.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods on Natural Language Processing*.

Kelly Peterson, Matt Hohensee, and Fei Xia. 2011. Email formality in the workplace: A case study on the enron corpus. In *Proceedings of the Workshop on Languages in Social Media*. Association for Computational Linguistics, pages 86–95.

Ye Qi, Devendra Sachan, Matthieu Felix, Sarguna Padmanabhan, and Graham Neubig. 2018. When and why are pre-trained word embeddings useful for neural machine translation? 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 529–535.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732* .

Sudha Rao and Hal Daumé III. 2018. Learning to ask good questions: Ranking clarification questions using neural expected value of perfect information. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1.

Sudha Rao and Joel Tetreault. 2018. Dear sir or madam, may i introduce the gyafc dataset: Corpus, benchmarks and metrics for formality style transfer. 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 129–140.

Ehud Reiter, Somayajulu Sripada, Jim Hunter, Jin Yu, and Ian Davy. 2005. Choosing words in computer-generated weather forecasts. *Artificial Intelligence* 167(1-2):137–169.

Nils Reiter and Anette Frank. 2010. Identifying generic noun phrases. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 40–49.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE, pages 1179–1195.

Philip Resnik. 1995. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th international joint conference on Artificial intelligence-Volume 1*. Morgan Kaufmann Publishers Inc., pages 448–453.

Philip Stuart Resnik. 1993. Selection and information: a class-based approach to lexical relationships. *IRCS Technical Reports Series* page 200.

Salvatore Romeo, Giovanni Da San Martino, Alberto Barrón-Cedeno, Alessandro Moschitti, Yonatan Belinkov, Wei-Ning Hsu, Yu Zhang, Mitra Mohtarami, and James Glass. 2016. Neural attention for learning to rank questions in community question answering. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. pages 1734–1745.

Barak Rosenshine, Carla Meister, and Saul Chapman. 1996. Teaching students to generate questions: A review of the intervention studies. *Review of educational research* 66(2):181–221.

Vasile Rus, Paul Piwek, Svetlana Stoyanchev, Brendan Wyse, Mihai Lintean, and Cristian Moldovan. 2011. Question generation shared task and evaluation challenge: Status report. In *Proceedings of the 13th European Workshop on Natural Language Generation*. Association for Computational Linguistics, pages 318–320.

Vasile Rus, Brendan Wyse, Paul Piwek, Mihai Lintean, Svetlana Stoyanchev, and Cristian Moldovan. 2010. The first question generation shared task evaluation challenge. In *Proceedings of the 6th International Natural Language Generation Conference*. Association for Computational Linguistics, pages 251–257.

Mrinmaya Sachan and Eric Xing. 2018. Self-training for jointly learning to ask and answer questions. 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 629–640.

Cicero D Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*. pages 1818–1826.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Controlling politeness in neural machine translation via side constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 35–40.

Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. 2016a. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 588–598.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016b. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Association for the Advancement of Artificial Intelligence*. volume 16, pages 3776–3784.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Association for the Advancement of Artificial Intelligence*. pages 3295–3301.

Fadi Abu Sheikha and Diana Inkpen. 2010. Automatic classification of documents by formality. In *Natural Language Processing and Knowledge Engineering (NLP-KE), 2010 International Conference on*. IEEE, pages 1–5.

Fadi Abu Sheikha and Diana Inkpen. 2011. Generation of formal and informal sentences. In *Proceedings of the 13th European Workshop on Natural Language Generation*. Association for Computational Linguistics, pages 187–193.

Nathaniel J Smith, Noah Goodman, and Michael Frank. 2013. Learning and using language via recursive pragmatic reasoning about other agents. In *Advances in neural information processing systems*. pages 3039–3047.

Noah A Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pages 354–362.

Svetlana Stoyanchev, Alex Liu, and Julia Hirschberg. 2014. Towards natural clarification questions in dialogue systems. In *AISB symposium on questions, discourse and dialogue*. volume 20.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Duyu Tang, Nan Duan, Zhao Yan, Zhirui Zhang, Yibo Sun, Shujie Liu, Yuanhua Lv, and Ming Zhou. 2018. Learning to collaborate for question answering and asking. 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 1564–1574.

Sean Trott, Manfred Eppe, and Jerome Feldman. 2016. Recognizing intention from natural language: clarification dialog and construction grammar. In *Workshop on Communicating Intentions in Human–Robot Interaction*.

Lucy Vanderwende. 2008. The importance of being important: Question generation. In *Proceedings of the 1st Workshop on the Question Generation Shared Task Evaluation Challenge, Arlington, VA*.

Tong Wang, Ping Chen, John Rochford, and Jipeng Qiang. 2016. Text simplification using neural machine translation. In *Association for the Advancement of Artificial Intelligence*. pages 4270–4271.

Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2013. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*. pages 404–413.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4):229–256.

Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation* 1(2):270–280.

Shuly Wintner, Shachar Mirkin, Lucia Specia, Ella Rabinovich, and Raj Nath Patel. 2017. Personalized machine translation: Preserving original author traits. In *European Chapter of the Association for Computational Linguistics*. pages 1074–1084.

Sander Wubben, Antal Van Den Bosch, and Emiel Krahmer. 2012. Sentence simplification by monolingual machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. Association for Computational Linguistics, pages 1015–1024.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics* 4:401–415.

Wei Xu, Alan Ritter, Bill Dolan, Ralph Grishman, and Colin Cherry. 2012. Paraphrasing for style. *Proceedings of International Conference on Computational Linguistics 2012* pages 2899–2914.

Hayahide Yamagishi, Shin Kanouchi, Takayuki Sato, and Mamoru Komachi. 2016. Controlling the voice of a sentence in japanese-to-english neural machine translation. In *Proceedings of the 3rd Workshop on Asian Translation (WAT2016)*. pages 203–210.

Jun Yin, Xin Jiang, Zhengdong Lu, Lifeng Shang, Hang Li, and Xiaoming Li. 2016. Neural generative question answering. In *North American Association of Computational Linguistics*.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE* 101(5):1160–1179.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. SeqGAN: Sequence generative adversarial nets with policy gradient. In *arxiv*.

Xingdi Yuan, Tong Wang, Caglar Gulcehre, Alessandro Sordoni, Philip Bachman, Saizheng Zhang, Sandeep Subramanian, and Adam Trischler. 2017. Machine comprehension by text-to-text neural question generation. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*. pages 15–25.

Zhicheng Zheng, Xiance Si, Edward Chang, and Xiaoyan Zhu. 2011. K2q: Generating natural language questions from keywords with user refinements. In *Proceedings of 5th International Joint Conference on Natural Language Processing*. pages 947–955.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A monolingual tree-based translation model for sentence simplification. In *Proceedings of the 23rd international conference on computational linguistics*. Association for Computational Linguistics, pages 1353–1361.