Towards Better Recommendation Explainability Evaluation for Conversational Recommender Systems

By

Joseph May

A thesis submitted in partial fulfillment

of the requirements for the degree of

MASTER OF SCIENCE

in

Computer Science

Middle Tennessee State University

May 2024

Thesis Committee:

Dr. Khem Poudel

Dr. Joshua L. Phillips

Dr. Jaishree Ranganathan

ACKNOWLEDGEMENTS

I would first like to thank my wife, Lori, who made me stick with it. Thanks to Dr. Poudel for getting me into his research group, and for being patient as I switched topics and made slow progress. Thanks to the reviewers who suggested edits and improved my paper. My thanks to the thesis committee, who helped me improve this document, and grow my knowledge. Thanks to Dr. Sarkar who asked me to be a GA which allowed me to finance my education. And a final thanks to my parents who got me here in the first place. Thanks all, the help was greatly appreciated!

ABSTRACT

This study focuses on Conversational Recommender Systems (CRS) and proposes a method for classifying recommendations as good or bad. Traditional conversational recommendation metrics like BLEU, ROGUE, and METEOR are not sophisticated enough to assess recommendation quality. A shift towards different metrics is needed to assess recommendation quality. Eight quality factors, length, readability, repetition, word importance, polarity, subjectivity, grammar, and feature appearance are proposed to be more relevant, explainable, and impactful metrics to assess conversational recommendation quality. Towards that end, three different neural networks are created using GPT2, GPT-NEO, and t5 as base models that embed a conversational recommendation and factor in the eight aforementioned quality factors as inputs to a linear residual network architecture to classify recommendations. The GPT-NEO model achieves the highest average prediction accuracy at 83%, GPT2 has an average accuracy of 78%, and t5 74%. Individual Conditional Expectation analysis shows that grammar, feature appearance, and repetition are the most impactful quality factors. A Shapley value analysis shows each factor can push model predictions toward bad or good classes for all three models. The 8 quality factors assess recommendation quality more meaningfully, accurately, and contextually than current standard methods.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I.

INTRODUCTION

Conversational recommender systems (CRS) are recommendation engines that determine what to recommend to a user based on a conversation [1]. They differ from search engines and other recommender systems in the primary mode of interaction, a conversation. A conversation takes place through multiple interactions of an information seeker, and the system. At each turn the user may express a desire (an item to search for), or introduce specifications (a color, a quantity, a price point etc), ask for more information, or ask the system for a recommendation. The main benefit of a conversational interface is that it lends itself to exploration [2, 3, 1, 4, 5, 6, 7]. If a user only has a vague sense of what they want, they can give the system what they know, and allow the system to continue prompting them until the system has a recommendation to suit their needs.

CRS are comprised of two main components, a conversation and a recommendation module [3, 8]. The conversation module manages the conversation. The goal of the conversation is to get the most information from the user at each turn of the conversation so that the conversation can conclude as soon as possible with the user receiving a recommendation [8, 9], a visual representation of this may be seen in Figure 1. The recommender engine oversees recommending items to the user. This can be done by searching for items and item parameters over the systems knowledge base, database, or over the internet. A CRS is designed to recommend an item at some confidence threshold. Getting more user information such as price range, color, size, etc, increases the likelihood that the recommendation is good, but takes more time. This is known as the dilemma of exploration versus exploitation [10, 11, 8], where the system must balance learning more about user preferences and exploiting that information to recommend an item.

An overarching concern for CRS is dialog strategy, which are the rules specifying how the speaker and system interact with each other. The goal of a CRS is to recommend an item

to the user as quickly and accurately as possible [1]. Consequently, conversational strategies are rules meant to swiftly facilitate the end of user interaction with the system. The longer the system takes to recommend an item reduces the overall utility of the CRS. A CRS that is interrogative and needs to exhaustively interact with users to recommend an item will drive users to other services or force users to lookup recommendations by themselves. This expresses the exploration versus exploitation dilemma all recommendation systems face. Four different paradigms exist to direct the flow of the conversation [3]. Each dialog strategy offers differing levels of conversational complexity and may factor in user profiling.

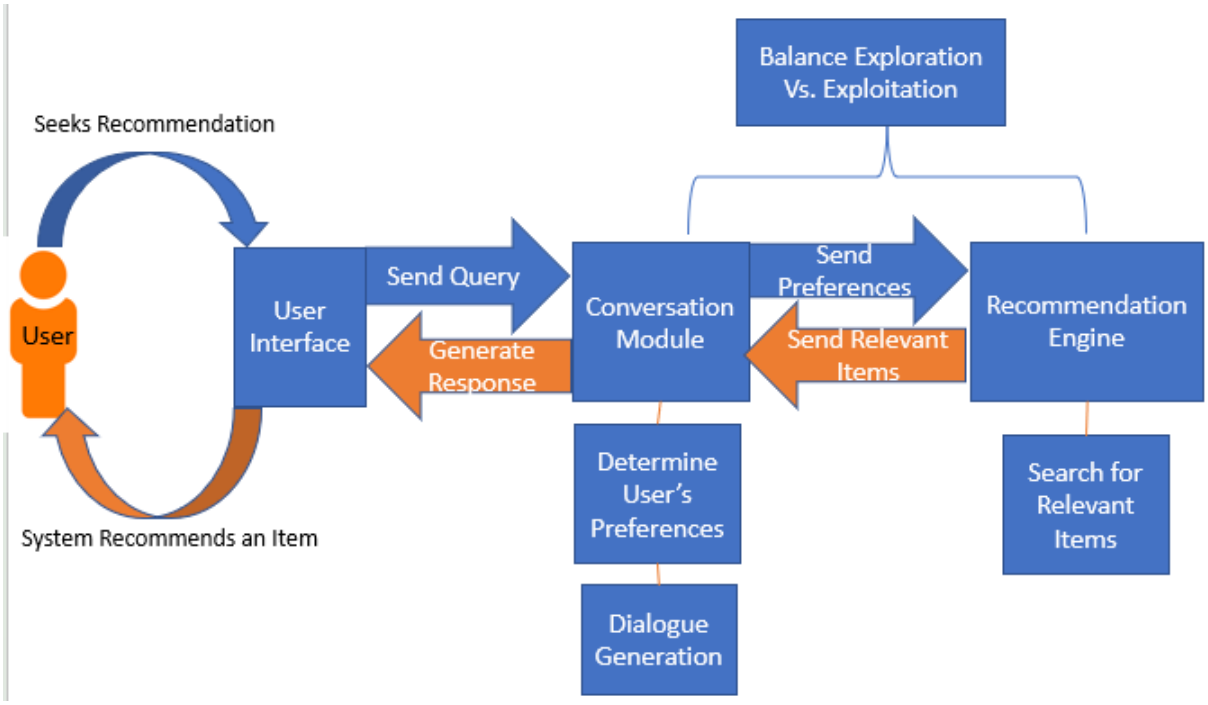

Figure 1: A high level diagram depicting the workflow of a Conversational Recommender System.

The system active user passive (SAUP) model has the system ask the user questions, and the user is meant to only respond to direct questions from the system. Excluding the original request for a recommendation, there is to be no initiation or interaction from the user without the prompt of the system [3]. Second is the system active and user engage (SAUE) model, which is like SAUP except that the user can "chit-chat" with the system. In this way, the

system mimics more human-like conversation through small talk-type interactions [3].Third is the system active and active user (SAUA) model. Here, the system can ask questions of the user but the user can also interrupt system processes to change preferences or narrow a search before the system prompts the user for more information [3]. Lastly, there is an active user and passive system (AUPS). This model is akin to a search engine or voice assistants like Siri. The system only acts when the user has given a command and has little to no initiative to recommend outside of direct prompting. AUPS is the most user-intensive and least conversationally capable of the four system paradigms [3].

The effectiveness of CRS depends on the quality of explanations for each recommendation. There exists a gap in assessing the quality of recommendations in a conversational setting to determine how well the recommendation met the needs the user expressed throughout the conversation [11, 1, 12, 13, 4, 14]. Current metrics primarily assess if the recommendation was successful, if it matches a desired format, or the speed of the conversation, and lack the ability to differentiate between high-quality effective explanations that make the user confident in the recommendation. The project aims to develop a network capable of classifying conversational movie recommendations by utilizing 8 quality factors to assess conversational quality, readability, repetition, grammar, feature appearnce, polarity, subjectivity, word importance, and length.

Benefits of Explainability

The Defense Advanced Research Projects Agency (DARPA) identified a need for explainable AI (XAI) in 2017 [15]. XAI aims to make AI systems understandable in how they work, why they work, and how they behave across various situations and input states. The European Union enacted the General Data Protection Regulation which established the rights of EU citizens about how AI systems reach their decisions. This law precipitated the FAccT conference to research and implement the philosophical tenets of XAI. FAcct was concerned with 1. Fairness- to be equitable, 2. Responsibility- to justify the actions of the

system, and 3. Transparency - decisions should be transparent to people who use and are affected by the system [15]. Vultureanu-Albisi and Badica credit Facebook with one of the first efforts to incorporate explainability into a recommender system with the addition of a because you are friends with tab when recommending new friends to add [15]. It has been shown that incorporating explainability increases user satisfaction, trust, and efficiency [2].

An explanation is a statement that clarifies the nature of something. Three important questions for XAI are what needs to be explained, how much must be explained, and who should be able to understand the explanation. *Interpretability* can be defined as providing meaning that can be understood by someone [15]. *Explainability* refers to the ability and degree that an action of the decision-maker can be interpreted by a user [15]. The degree to which something is explainable and interpretable is domain-dependent. Interpretability and explainability are grouped into global and local levels. The global level features total transparency where all workings are explainable and explanations are provided for as many user types as possible. The local level is more common and only explains select portions of the decision-making process [15]. Explanations serve different purposes for different audiences. Not all system processes will be interpretable to all audiences, as some processes are too technical/too time-consuming to useful for the average user.

Recommender systems must overcome the *cold start problem*, and the *concurrent use problem* [16, 17, 1, 12, 18, 15]. Recommendations from strangers tend to have little impact. All recommendation systems start as strangers to a user. This means that the system needs to be good for users to continue to use the service. This is difficult as the system will have the hardest time recommending items for new users due to the cold start problem. The system outputting an explanation can help to justify a recommendation and mimic how humans recommend content to one another and can help the system mitigate missteps. As an example, the output, “Try watching The Office” is not as convincing as the output, “Try watching The Office. It is a comedy series that is well-rated by fans and features Steve Carell,

an actor whose movies you have watched before." The second output is more convincing and gives the user more reason to trust the system. Furthermore, an explanation can help the user identify where the system has gone wrong. For example, "Try watching The Office it is a comedy series," can help a user identify and provide feedback that they are not interested in comedy shows, and would rather watch another genre [19]. Justification mainly serves users of a recommender system and serves to explain why a recommendation is relevant to them, and give them the tools to understand why a recommendation was given and change how they interact with the system to alter its personalization towards them [15].

CRS can be improved both by users and system designers when explainability is incorporated. Users can understand how their actions affect the system adjust accordingly to get better results by doing things like giving ratings to items, adjusting settings, or by sending feedback to the company. System designers can incorporate user feedback to add features users want or identify when and where the recommendation method needs to be tweaked [15]. User preferences can drift over time, or users may want a more varied experience. Even if someone loves action-comedy movies, they probably don't want to just watch action-comedies and would like the system to recommend other types of movies. User feedback can let designers know that the amount of variety recommended for a particular type of user should be increased. In this way, explainability serves product users and product designers. Current assessment metrics such as BLEU, perplexity, and average entity score are interpretable and explainable but fail to asses a conversational recommendation on terms that align with human preference. The eight quality factors remain interpretable and explainable and also overlap with what humans want to see in a recommendation explanation.

CHAPTER II.

Background

Explainability in CRS

Recommender systems provide a single-turn experience. The user will ask for something, and the system will output a recommendation. If the answer is not to the user's liking, they must refine their search parameters and try again. Users engage with recommender systems when they are unsure of what they want, otherwise the user could just search for an item specifically. A typical interaction pattern results from this dynamic, where a user submits an ill-formed or generic query, and receives a generic or unhelpful system response. The user then needs to assess what went wrong and try again. Traditional recommender systems offer few ways to incorporate user feedback [19]. Conversely, because a CRS is designed with conversation in mind, the recommendation process is iterative and occurs over the space of multiple dialogue interactions where the user can provide explicit desires and feedback to guide the system [19]. Because a CRS is continually seeking user interaction, the burden is lessened from the user, who can just inform when and how much they like system output. The benefits of explainability, as outlined above are clear, and furthermore, Guo et al. show that explainability is highly correlated with the success of a CRS [8]. However, designing a CRS is a challenging task. Chen et al. [19] identified three key objectives for a CRS:

1. Integrate user feedback and steadily improve recommendation accuracy.
2. Explain recommendations.
3. Stay within the bounds of user feedback.

Guo et al. [8] point out that objective three means the system must tailor responses to a specific user, while objectives 1 and 2 require the system to be generalized. Recommendations must be explained and justified, as must the dialog output of the system. This means that to incorporate explainability for designer and regulator stakeholders the system must

be able to explain why it said what it said in addition to explaining its recommendation. This is a substantial increase in the number of explanations generated by the system when compared to traditional recommender systems.

Factors Impacting the Quality of an Explanation

Many factors affect the overall quality of an explanation, this paper examines eight of the most commonly adopted factors [17, 2, 19, 20, 8, 21, 12, 15, 14].

Length: The length of an explanation can impact its effectiveness. Lengthier explanations may exhibit the properties of being more verbose, and more repetitive, and may contain more words, but mean little. Shorter explanations may be accurate but can be less persuasive as not enough justification has been done to gain the user's trust that the recommendation is good. The length of an explanation is measured the same as by Wen et al. [14] which is by recording the number of words in the recommendation after removing stop words.

Readability: The readability of a text determines how easy the passage is to understand. This will be measured by the Flesch-Kincaid readability test [22]. There are two versions of the Flesch-Kincaid test, reading ease and grade level. Flesch-Kincaid grade level tests are not used in this document, only reading ease. Flesch-Kincaid reading ease is an automated formula to determine how hard material is to read. The formula has 3 variables, total words, total sentences, total syllables, and 3 constants, 206.835 the maximum possible score, 1.015 a weighting factor adjusting the score based on sentence length, and 84.6 which is another weighting factor adjusting the average number of syllables per word [22]. The formula is thus:

$$206.835 - 1.015 * (\frac{TotalWords}{TotalSentences}) - 84.6 * (\frac{TotalSyllables}{TotalWords})$$

In the Flesch reading ease test, higher scores indicate that the material is easier to read.

Word Importance: The word importance of an explanation is a measure of how important each word in the explanation was. Recommendations with low word importance scores tend to be filled with less useful information and content not relevant to the rec-

ommendation topic. Recommendations with high word importance scores are more likely to be relevant to the current recommendation topic. Word importance is calculated by taking the sum of the word importance scores of each word in a recommendation. The individual word importance scores for each word are determined by term-frequency inverse document frequency (TF-IDF). Term frequency (TF) measures how often a word appears in a recommendation explanation and is calculated by taking the total times a term was used in the recommendation explanation over the total number of terms in a recommendation explanation [23]. Inverse document frequency (IDF) measures how important a word is across multiple recommendation explanations and is calculated by this formula: [23] $log(\frac{\text{Total number of recommendation explanations}}{\text{Number of recommendation explanations containing term T}} + 1$

TF-IDF combines TF and IDF to score each word in a recommendation. Words that are common in an explanation but not across multiple explanations will have high TF-IDF scores, words common across multiple explanations will have low TF-IDF values. Higher TF-IDF scores indicate a word is important in the current recommendation [23].

Repetition: Repetition refers to how many duplicate words an explanation has. Repetitive sentences tend towards redundancy, while more varied sentences tend to be more engaging. Repetition is calculated by counting the number of repeated words once stop words have been removed from an explanation [14].

Subjectivity: Subjectivity reflects if the recommendation contains personal opinions, emotion, and/or judgment [11, 14]. Recommendations that list facts do not have the same success rate of those that offer extra opinions on the subject matter. The ideal amount of subjectivity in a recommendation is not known. Subjectivity is calculated using textblob.

Polarity: The polarity of an explanation indicates the tone of the sentence and is expressed in as a positive or negative tone [11, 14]. The tonal output of the system must correspond to the certainty of its response. Confident recommendations should receive stronger support than less confident ones. Polarity is calculated using textblob.

Grammatical Correctness and Conventionality: Incorrect grammar within a response negatively affects the readability and persuasiveness of an explanation. Explanations should feature proper grammar and follow common conventions. Following Wen et al. [14], this is evaluated by using the Python language tool.

Feature Appearance: The relevance of an explanation reflects how much of an explanation encompasses the core components of the item it is meant to explain. The more details provided that match an item's features indicates a higher degree of relevance. Relevance is determined by examining the web of words and concepts within an explanation both in semantic meaning and in the overall query context and is calculated by taking the cosine similarity between the seeker's request and the recommendation response. Explanations with relevant feature descriptors for an item that pair with a user's requests are likely to be relevant explanations. System explanation must include the primary user's request, and receive higher relevancy scores when more conditions are included [14]. A summary of all quality factor descriptions may be seen in Table 1.

Conversational Metrics

Conversational metrics are the methods by which a CRS's conversational output may be rated. These metrics focus on different areas like grammatical correctness, sentence complexity, informativeness, topicality etc. What follows will be a description of conversational metrics found in the literature. An explainability score needs to incorporate aspects from all these domains to accurately score the explainability of CRS output. Other methods exist and are often applied in online evaluations through crowdsourcing. We focus on an offline evaluation for efficiency, cost, and reproducibility [2, 12].

Recall, Precision, and F scores [24, 10, 11, 1, 25, 4, 18, 26] are used in conversational evaluation as well as recommendation evalaution. A CRS must be able to identify important parts of a sentence. Words need to be classified into categories like location and person in a process known as named entity recognition. Important categories are named beforehand

Table 1: Quality Factor Summary

Feature Name	Feature Description	Calculation Method
Length	How long a recommendation is after stop words have been removed	Tokenize the recommendation, and count the number of words remaining.
Readability	How easy a recommendation is to read.	Flesch-Kincaid Reading Ease test.
Word Importance	How impactful/meaningful a word is to a recommendation	Term Frequency Inverse Document Frequency (TFIDF) analysis.
Repetition	The number of repeated words in a recommendation after stop words have been removed.	Tokenize the conversation, and sum the count of repeated words in the recommendation.
Subjectivity	How much personal opinion, emotions, and judgement a recommendation contains	Textblob
Polarity	The tone of the recommendation	Textblob
Grammar	How much a recommendation follows grammar conventions in English.	PyspellChecker
Feature Appearance	The amount of item features (genre,cast,shape,color,price etc) mentioned in a recommendation in response to a user query, or in support of a recommendation.	Cosine similarity between user query and recommendation output.

and the model attempts to recognize when an entity of that type has shown up in a sentence. Recall measures how many correct entities have been recognized in the set of all true and false positives, precision is the number of correctly identified entities divided by the number of true positives in the set, and the f1 score is the harmonic mean between precision and recall. There are other natural language processing (NLP) tasks a CRS performs such as sentiment analysis, and keyword extraction. Recall, precision, and F score, may be used to evaluate how successful a CRS is at these tasks.

Feature Level Precision and Recall [2] evaluates explainability by comparing the features in a predicted sentence against real reviews. Feature precision (FP) is calculated by:

$FP_{u,i} = \frac{|S_{u,i} \cap T_{u,i}|}{|S_{u,i}|}$.

Feature recall (FR) is calculated by:

$FR_{u,i} = \frac{|S_{u,i} \cap T_{u,i}|}{|T_{u,i}|}$.

Above, (u,i) is an item pair, the predicted features are S_u, i and the real features are T_u, i.

Mean Explainability Precision and Recall [2] are the average of explainable precision and recall above, and represent the average explainability precision and recall. Mean explainable recall (MER) and mean explainable precision (MEP) are calculated by:

$MER = \frac{|N_u \cap M_u|}{N_u}$, $MEP = \frac{|N_u \cap M_U|}{|M_u|}$.

Above, u indicates a user, N_u us the set of items that can be recommended to a user, and M_u is the set of recommended items.

Average Entity Score measures how many entities are mentioned in a response on average [11]. The higher the average entity score the more entities are mentioned in a sentence. A user can only process so many recommendations at a time. If a CRS recommends a lot of items in a single pass, it can be too much information and reduce the usability of the system and muddle the clarity of system recommendations. Different CRS have different strategies for recommendation and will correspondingly aim for different average entity scores based on their recommendation strategy.

Vector Extrema calculates sentence level embeddings. Given the vectorized form of all the words in a sentence, take the most extreme value, and use that as a sentence-level embedder. The similarity between this sentence and possible sentences can be computed using the cosine distance between the extreme embedded sentence and the proposed sentence. This method will place common words closer to the origin while more unique words will be placed further from the origin inflating the importance of unique words [18]. This allows the comparison to place more emphasis on unique words which are more likely to hold

more semantic weight, than common words. As an example, take the sentence, “let’s go get the kids.” A model looking for word overlap would place a lot of importance on possible sentences sharing n-grams like, “let’s go get the car,” and, “let’s go get the food.” By basing the vectorization on distinct words, the effect of common words are reduced, so unique words like car, food, and kids have increased importance. The model can then compare the distance from kids as the ground truth to car, and food ignoring the overlap of, “let’s go get,” to determine how good a match the proposed sentences are to the ground truth.

MAUDE is a novel method of dialogue evaluation where a model uses pre-trained language models to extract dialogue representations that has achieved a high correlation with human judgments and does not require true responses for comparison [27]. This method shows promise due to its high level of automation and correlation with human judgment. MAUDE had a stronger correlation with calibrated human judgments than DistilBERT due to MAUDE’s strong correlation with interestingness and engagingness, which measure how interesting and enjoyable the conversation was when talking to a CRS using MAUDE [27]. A common issue with dialogue evaluation is that evaluation often requires human references, and do not generalize to unfamiliar domains. MAUDE both generalizes well and does not require human annotation to asses dialogue quality. To our knowledge, this method is not widely used but has the potential to outperform more standard metrics like BLEU.

Unique sentence ratio is a procedure where the sentences of a model’s output are analyzed [2, 20]. This is used to evaluate the results of generative systems. The sentences a CRS outputs will be compared to the training data to determine the degree to which the model is generating its sentences or parroting training data. This may or may not include judgments on the semantic worth of model output as well [20].

Distinct n-gram evaluation looks at the number of unique n-grams in a sentence. A monogram is a single word, a bigram is two words next to each other, a trigram is three words in a sentence next to each other, and an n-gram is n number of words next to each

other in a sentence. In the sentence, “I am hungry today,” there are four monograms (I, am, hungry, today), 3 bigrams (I am, am hungry, hungry today), and 2 trigrams (I am hungry, am hungry today). N-gram evaluation is typically done in the range of 1-4 n-grams [26]. Sentences that have a low number of unique n-grams are sentences that reuse words. There is nothing strictly wrong with repeated words, but repetition can often lead to sentences that sound strange. Typically speaking sentences that have a higher diversity of words are preferred over those with lower word diversity.

Perplexity is the probability a model assigns a word to fill in a blank [11]. This measures how well a system understands word choice. Every word in the system’s vocabulary will be assigned a probability rating based on the frequency of use and context. A model with high perplexity is unable to predict the next word of a sentence due to its word probability distribution being incorrect. A model with low perplexity can accurately predict the next word in a sentence. Perplexity is strictly a measurement of the likelihood a word can fit into a blank space in a sentence, it does not account for any semantics in a sentence. In the sentence, “Jack the Ripper was a ____,” probable responses to fill in the blank could be person or man. It is unlikely just using perplexity that a model could fill in more semantically correct words such as killer or murderer because killer and murderer are generally speaking not probable words in many sentences compared to person or man. Because perplexity measures probability and not any semantic meaning, the model can output correct sentences that are often not very meaningful or contextual.

Cosine similarity is used to assign class labels in the experiment. Cosine similarity measures the similarity between two vectors in a multi-dimensional space. It calculates the cosine of the angle between the vectors, providing a measure of how closely aligned the vectors are. The cosine similarity ranges from -1 to 1. The closer to 1 the angle is the more similar the vectors are, 0 implies perpendicularity and no similarity, and -1 suggests the vectors are dissimilar. It is useful when the magnitude of the vectors is not crucial for the

task, and the focus is on the direction or orientation of the vectors in the vector space.

BLEU is a word overlap metric that looks for shared n-grams between the ground truth and possible responses [21, 18, 28]. This metric assumes that there is a single ground truth response to compare to. BLEU can be set to look for different numbers of word overlaps such as 2,3, or 4. As an example, the ground truth response may be, "*The Bridge on the River Kwai,*" and two proposed responses are, "*The Bridge on the Mekong River,*" and, *"The Kwai river's bridge.*" BLEU looks for shared n-grams so the first example will have a higher BLEU score because it shares the n-grams, *"The Bridge on the"*, while the second answer shares only monograms with the ground truth. The more words that are shared between responses increase the likelihood that the results share a similar meaning, but that does not mean that overlap is the same as the actual semantic meaning of a sentence. Even though the first proposed answer has high overlap it is further away from the semantic meaning of the ground truth because it is talking about the Mekong River, and not the river Kwai.

METEOR is another word overlap method that was created in part to address the shortcomings of BLEU. METEOR will generate an alignment between the text and a proposed response. Then the METEOR score will be calculated as the mean of precision and recall between a sample sentence and the ground truth [29, 30]. If we consider the sentence the rat sat on the step. The alignment will be a set of words that do correspond to adjacent words in the reference. In the example, the word "the" can correspond to either of the "the" in the sentence. The alignment will try to place words where the number of word chunks is the lowest. In the ideal scenario, the ground truth and the proposed sentence match, and each word can be aligned in its original place. When this is not the case, the f score is used to determine how likely a match is and is given by the formula: $\frac{10*Precision*Recall}{Recall+(9*Precision)}$.

ROUGE is an F-measure based on the longest common subsequence between a proposed sentence and the ground truth. The longest common subsequence is a set of words that appear in two sentences in the same order. The longest common subsequence is similar to

the n-gram matching in BLEU, but the subsequences do not have to be touching as is the case with n-grams [31, 18]. Consider the two sample sentences, “the sandwich in the fridge,” and, “the sandwich and the chips please.” The longest common subsequence in these two sentences is, “the sandwich the” which is the set of words shared between both sentences, even though the words are not adjacent to one another. Then the precision and recall are calculated and the f1 score is used to determine the degree to which the sentences match.

BLEU, ROGUE, and METEOR are commonly used to assess the quality of machine-generated output, and can capture measurements on quality, but do not capture the full semantic meaning of words, and can fail to understand words in context. These methods can struggle with tones such as sarcasm and irony and additionally have issues with paraphrases which mean the same semantically, but have little to no overlap in words with the known ground truth [18]. It is possible to augment the ability of these metrics to handle paraphrases, but this requires the addition of more ground truths to accommodate for paraphrases of any given sentence, which is time and labor-intensive and may not yield significantly better results. Other metrics such as precision, recall, and perplexity measure different aspects of model output but are not directly tied to the conversational recommendation the way the 8 quality factors are. Perplexity, for instance, measures how well a network can model correct word use, but that is not a direct assessment of how good a recommendation is. Many of the offline metrics used in CRS evaluation have the quality of capturing and assessing different aspects of language generation but are not directly related to whether or not the explanation is good. This can most clearly be seen in the divide between offline metrics, and online evaluations where humans grade recommendations on factors like the 8 quality metrics [11, 12, 13, 14, 9], both because manually calculating BLEU scores or finding the perplexity value makes little sense for survey participants and because humans prefer explanations that have strong feature appearance and word importance over models that have higher entity scores. A table of common evaluation metrics may be seen in Table 2.

Table 2: Conversational Evaluation Metrics Summary

Metric name	Used In
BLEU	[20],[32],[21],[12],[18],[33]
ROGUE	[11],[12],[18]
METEOR	[18]
Vector Extrema	[18]
N-gram	[11],[12],[33] [6]
Precision/Recall	[12],[33],[34],[6]
Perplexity	[32],[11],[21],[12]

Deep learning methods are more commonly used to capture semantic meaning, and rely on word embeddings to do so. Word embeddings are vector representations of words in a continuous vector space, which are intended to capture semantic relationships and contextual information. These embeddings serve as a fundamental component in NLP tasks, providing a dense numerical representation of words that facilitates machine learning models in understanding and processing language [35]. Word embeddings encode semantic relationships between words based on their context in large corpora. Words with similar meanings or usage patterns have similar vector representations. This allows models to understand and leverage the semantic nuances of language, capturing similarities and differences between words [35]. Each dimension in the embedding space represents a specific linguistic feature, enabling models to understand the context in which a word appears. This contextual awareness is crucial for tasks such as sentiment analysis, named entity recognition, and machine translation. Furthermore, word embeddings reduce the dimensionality of the feature space by representing words in a continuous vector space with lower dimensions, models can effectively learn and generalize from limited amounts of data [35]. Word embeddings and the transformer architecture have revolutionized NLP. The Transformer architecture is a neural network architecture introduced in the paper “Attention is All You Need” by Vaswani et al. [36]. The transformer uses self-attention to weigh

different words in a sequence differently when making predictions, which allows the model to capture dependencies between words in a sequence, allowing the model to understand semantic meanings in ways that BLEU, ROGUE, and METEOR cannot. The transformer relies on positional encodings which are added to the words in a sequence to give the model tools to understand the structure and position of inputs in a sequence [36]. Transformers are parallelizable and feed forward which makes them a fast architecture as well. The Transformer architecture has proven to be highly effective for various natural language processing tasks like machine translation and summarization, and in particular BERT [37] and BART [38] transformer models are used in this experiment.

There are many conversational metrics used to evaluate CRS. These metrics encompass different aspects such as grammatical correctness, sentence complexity, informativeness, and topicality. The goal is to develop an explainability score that incorporates elements from these domains to accurately rate the explainability of CRS output. While other evaluation methods, such as online evaluations through crowdsourcing, exist, the focus here is on offline evaluation for efficiency, cost-effectiveness, and reproducibility. These metrics capture various aspects of CRS output quality and explainability, including semantic meaning, word diversity, and alignment with human judgments.

Previous Work

Bidirectional Encoder Representations from Transformers (BERT), was introduced in "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," Devlin et al. [37]. BERT was created to address the limitations of traditional language models that processed text in a directional fashion (left to right, or right to left only) [37]. Uni-directional models struggled to capture contextual information and relationships between words that were distant from each other in a sentence. BERT aimed to overcome this limitation by pre-training a deep bidirectional transformer model on a large corpus of text. This pre-training methodology helped the model to understand the meaning of words

in the context of the entire sentence, leading to more contextually rich and semantically accurate representations [37]. BERT is used to embed conversations when calculating the feature appearance scores in this work. Bidirectional and Auto-Regressive Transformers (BART), is a sequence-to-sequence model introduced in "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension." by Lewis et al. [38], and is used to summarize recommendation explanations. BART in a denoising autoencoder fashion. The model is designed to handle a variety of natural language processing tasks, including natural language generation, translation, and comprehension. BART incorporates both bidirectional training like BERT but also auto-regressive training objectives. BART is trained by corrupting input sentences and then reconstructing the original sentence [38]. This denoising objective encourages the model to learn robust representations by understanding the relationships between different parts of the input. Bart summarizes a conversation and is used for calculating the feature appearance score.

The paper Measuring "Why" in Recommender Systems: a Comprehensive Survey on the Evaluation of Explainable Recommendation [2] provides a comprehnsive survey of methods for evaluating explainable recommendation systems. It focuses on understanding why a recommendation is made, which the paper identifies as an essential component for user trust and acceptance of a recommendation. The authors discuss various evaluation techniques, including both user-centric and system-centric approaches [2]. The authors categorize the methods into intrinsic and extrinsic evaluation and discuss the strengths and limitations of each approach. The paper takes time to emphasize the importance of considering different dimensions of explanation quality, such as informativeness, transparency, and user satisfaction in the evaluation process as one dimension can only encompass and evaluate a particular aspect of an explanation [2]. Multiple dimensions are needed to holistically determine the quality of an explanation.

Towards Explainable Conversational Recommender Systems by Guo et al. [8]. ad-

dresses the challenge of providing explanations in conversational recommender systems (CRS). It proposes a framework that integrates recommendation generation with explanation generation. This dual generation framework allows for natural interactions between the system and users. The authors introduce a two-step process, where the system first generates a recommendation and then constructs an explanation based on the user's context and preferences. This approach aims to enhance user understanding and trust in the system's recommendations [8]. The paper highlights the potential of this framework in improving user satisfaction and engagement in CRS, which is a core goal of this work as well.

ExpScore: Learning Metrics for Recommendation Explanation by Wen et al. [14]. introduces a novel approach to learning evaluation metrics for recommendation explanations. The authors propose a method that leverages reinforcement learning to train a metric that assesses the quality of explanations, named ExpScore. ExpScore is trained to maximize the correlation between human judgments and the model's scores for explanation quality [14]. Similar to measuring why in recommender systems, this paper also highlights the need for multiple recommendation factor inclusion for classification purposes. The paper demonstrates the effectiveness of ExpScore in both automated and human evaluations, showing that it outperforms existing metrics. These papers provide valuable contributions to the field of recommendation systems by offering a data-driven and adaptable method for evaluating the quality of explanation generation by incorporating different facets of explanation quality indicators into evaluation procedures which this project aims to emulate.

Y. Zhang uses a SAUP conversational strategy and a collaborative recommendation strategy [9]. This is the paper that most defines the SAUP model. The authors used Amazon purchase datasets to test their work. The dataset consists of millions of customers and products containing reviews, product descriptions, and multi-level product categories. The authors used four subcollections in the dataset which were electronics, CD & Vinyl, Kindle Store, and cell Phones. Using previous work, the authors extract aspect value pairs from each

subcollection and use the pairs to turn reviews into conversations for the CRS. The authors create a multi-memory network architecture. Items are represented as vectors based on item descriptions from their subcollection and are also based on the user's initial request, similar to what is done in this work. This embedding is combined with an attention mechanism to find relevant signals for memory and for search and question generation.

Y. Zhang et al. [9] main system evaluation metric is hit ratio @k to evaluate their model. This metric measures how many successful recommendations a model has achieved by the k turn of a conversation. They additionally use mean average precision, mean reciprocal rank, and normalized discounted cumulative gain to assess the performance of individual tasks like recommendation and question clarification. The authors found that the model performed better with high k levels, meaning the longer the conversation went on the better the model performed. This makes intuitive sense as the longer a conversation is the more information a model has to provide a recommendation. The model was able to more accurately predict user questions when more predictions were suggested to the user, and suggested questions were found to be chosen the longer a conversation went on [9]. Hit ratio@k is an excellent metric to measure overall system performance, but is not specific enough to provide insight as to why the system is working or why it fails under certain circumstances.

Another method to suggest items to users is utilizing knowledge graphs as K. Zhou et al. did [6]. This model uses a SAUE recommendation strategy and a hybrid of collaborative and content recommendation strategy. In this method, there were two knowledge graphs, one holding information on items a user may want, and one holding semantic information about words such as synonyms. When the user asked for something, the information embedded in the word graph would enable the CRS to understand what the user wanted, and the item graph would help the CRS recommend an item. By combining semantic meaning with item representation, the system can handle a more robust space of recommendations so long as it can understand what a user means and has information on that item in its graphs. Wong et al.

[39] used a similar method with a billion-scale knowledge graph. Their knowledge graph contained information about users, items, and conversations. User information was kept as a triple in the form of (user, has, tag). Item information was split into different categories. Seller information was kept as a triple in the form of (seller, has, item), and item categories were in the form of (item, belong to, category). Item properties were split into two categories one indicating an item had a particular feature (item, has, value) and another describing that feature (property, has value). Four triples were created for conversations, (user, created, session), (session, relate to, seller), (session, has, intention), and (intention, has, keywords). They pre-trained their model on a knowledge graph with 500 million entities and 6 billion triplets using Alibaba max-reduce. The authors compared their model's performance against 5 baseline models: Deep Cross, Wide deep, Deep Neural Network, Gradient Boosted Decision Tree, and Logistic regression across 10 datasets. Their method achieved an average improvement of 1.2% and performed better in sparse data environments which was expected due to the ability of the model to reference a large knowledge base that other models do not have access to. K. Zhou et al. [6] and Wong et al. [39] methods provide similar or higher levels of explainability than the methods proposed in this study but do require sufficiently sized external knowledge bases to provide context and explain recommendations.

W. Lei et al. noted that item recommendation involves tradeoffs between certainty that the recommendation is good and decreasing efficiency by clarifying with the user [5]. This model uses a SAUA conversation strategy and a hybrid utility collaborative recommendation strategy. The authors proposed an estimation action reflection (EAR) model of conversation, which is sort of a modified version of Y. Zhang et al's and their System Asks User Responds (SAUR) model [9]. The certainty that an item is good comes from past data and current dialog. A subproblem that stems from user history mining is handling users with few previous interactions / outdated user history (cold users) [4]. Using the most current user preferences is an advantage CRS have over traditional search tools [9]. Even though the user

may never have expressed interest in an item or feature before, they may have developed one because preferences change over time. K. Zhou et al. [7] also brought up the issue of how to deal with negative responses. A no might mean the recommendation is bad, or the user already has the item, or cannot afford it, and points out that the information gained from a negative response is much smaller than the information gained from a positive response and suggested a way to deal with this issue [7]. The authors devised a model that highly prioritizes both historical data and attribute preferences from the current conversation. A related issue is how to balance recommendation biases. Should the system weight items that are more popular, and items that are similar but less expensive, and how would that change based on previous user preferences [1]? These questions are currently being researched.

Krauth et al. showed that offline metrics correlate with online performance [13]. Krauth et al. also found that CRS research evaluation is often performed in experiments that are not replicable. The level of correlation between offline results and online performance is weak, and it remains unclear which metrics have the highest level of correlation, when optimizing for a metric is no longer useful, and which metrics most closely align with human judgment. Furthermore, certain recommendation algorithms perform worse as recommender systems scale up for large enterprise uses. This work is supported by Sinha et al. [18]. Word overlap measurements are quite common in the literature, but show a weak correlation with what users like. The authors found that word overlap metrics had a small positive correlation on the Twitter dataset and no correlation on the Ubuntu Dialogue Corpus. Due to the misalignment with human judgments, the authors concluded that word overlap metrics should be phased out as metrics that align more closely with user preferences are found.

Goldstein et al. [40] created the individual conditional expectation (ICE) algorithm to visualize the behavior of neural networks. ICE plots provide a visual representation of the effect of a single feature on the prediction of a model. ICE plots differ from partial dependence plots which represent the average effect of a feature across the entire dataset.

ICE plots show the effect for each individual instance. By plotting the predictions for multiple instances as separate lines on the same graph, ICE plots enable analysts to observe the variability in predictions across the feature space. This helps in understanding how the model's predictions change for different values of the feature of interest, thus providing insights into the model's behavior and highlighting potential nonlinearities or interactions. ICE plots offer a flexible and intuitive tool for interpreting complex machine learning models, and are used in this study to visualize the effect of each 8 quality factors on model predictions. Lundbeg et al. [41] present a framework for interpreting neural network predictions that they call SHapley Additive exPlanations (SHAP). SHAP values provide a unified approach to explaining the output of any model by attributing the prediction to each feature's contribution. The key idea is to utilize Shapley values from cooperative game theory, which assign a value to each feature based on its contribution to the prediction's deviation from the average. This framework ensures local accuracy, meaning the sum of feature attributions equals the difference between the prediction for a particular instance and the average prediction. Additionally, SHAP values offer consistency across different feature attribution methods, allowing for fair comparisons. By providing interpretable feature importance scores, SHAP values enable analysts to understand the model's decision-making process, identify influential features, and detect biases. SHAP is used in addition with ICE to explain the results of the model. The ICE analysis provides an in-depth local explanation of the model results, while the SHAP analysis helps to identify influential quality factors across the entire dataset, rather than an individual conversation.

Three base models are used in this work, GPT2 [42], GPT-NEO [43], and t5 [44]. Radford et al. [42] introduced the generative pretrained transformer 2 (GPT2) which is a large unsupervised language model trained on internet data. GPT2 is built on the transformer architecture from Vaswiani et al. [36], which allows the model to capture long range dependencies in data. GPT2 demonstrated high success at jobs it was not specifically trained

for. The model is highly influential and improved on past successful training technique breakthroughs. The model generalizes well to unseen data and tasks and was selected for this capability. Sid et al. [43] created the GPT-NEO (NEO) model. The NEO family of models are open-source attempts to create models with similar performance capabilities and behaviors of proprietary large language models like GPT2. As such, GPT2 and NEO exhibit many similar tendencies. NEO was selected due to its generalization and broad NLP capabilities. Raffel et al. [44] created the t5 model to explore the limits of transfer learning. Like GPT2 and NEO, t5 is a transformer model, however, unlike GPT2 and NEO t5 has a text-to-text objective instead of an auto-regressive objective. The t5 model performs well across a variety of NLP tasks, but may need more fine tuning than GPT2/NEO.

Base Datasets Used:

This paper uses an amalgamation of two well-known CRS datasets, E-redial [8] and INSPIRED [21]. E-Redial was created as part of an investigation into the necessity of explainability for Conversational Recommender Systems (CRS). The authors evaluated five widely used CRS datasets, ReDial, TG-ReDial, DuRecDial, INSPIRED, and OpenDialKG. 20 participants assessed system responses from sampled dialogues, providing evaluation metrics at the exchange level. The results indicated that existing datasets had relatively low-quality explanations, with issues such as lack of explanation, ambiguous recommended reasons, unrepresentative item descriptions, low effectiveness, efficiency, user satisfaction, trust, or willingness to accept recommendations in most dialogue turns. To enhance the explanation quality of CRS, the authors conducted a user study to identify characteristics of good explanations. Four main characteristics were identified: clear recommendation reason, representative item description, encouragement or personal opinion on recommended items, and being reasonable and contextual [8]. Manual and automatic methods were employed to rewrite low-quality explanations on the ReDial dataset, resulting in the creation of the Explainable Recommendation Dialogues (E-ReDial) dataset.

The INSPIRED dataset was developed by Hayati et al. [21] and is a collection of movie recommendation dialogues with extra annotations to allow for a model that pursues social strategies to build rapport with the seeker and increase recommendation acceptance. The authors used ParlAi to collect dialogue data, and crowd workers from Amazon Mechanical Turk. Conversations between participants were required to go for a 10-turn minimum, or until a recommendation was made. Conversations were annotated by human workers with linguistic backgrounds and portions of each conversation were tagged with strategies such as offering help, preference confirmation, and personal opinion.

CHAPTER III.

Methods

The github page with the code may be found here. The conversations from E-redial and INSPIRED are collected and combined into a single dataset along with the scores of each quality indicator into a text file with the following format: conversationID, seeker conversation, recommender conversation, length scores, readability scores, word-importance scores, repetition scores, subjectivity scores, polarity scores, grammar scores, feature appearance scores, and the whole conversation with the order preserved. Certain quality indicators are scored solely based on what the recommender has said and for convenience the seeker and recommender conversations are a record of the conversation with only that particular user's contributions, whereas the preserved conversation records the conversation as it happened with the addition of role tags between utterances such that the seeker's words are preceded by a "SEEKER:" tag and the recommender's words are preceded by a "RECOMMENDER:" tag. The data is stored in a CSV-like format where the delimiter is $|=$.

Target Creation

Neither E-redial nor INSPIRED datasets contain classifications of good and bad recommendations. Rather, both datasets contain conversations between a seeker and recommender, where the seeker is looking for a particular type of movie, and the recommender tries to recommend suitable movies for the seeker. Target labels needed to be created for the model to use to classify each conversation. The target labels are based on the scores assigned for each of the eight quality indicators for a good recommendation, where each score is based solely on the recommendation messages unless otherwise mentioned.

The length score for each conversation is a value that is scaled by the length of conversations in the dataset after stop words have been removed. The stop words used are the base stop words in the Python NLTK library, punctuation marks, angle brackets, square brackets, and curly braces. Once the stop words have been removed, the standard deviation

for the training dataset is 93.112 words and the average conversation length is 216.429 words. The test dataset has a standard deviation of 88.557 words and an average length of 195.451 words. To score a conversation the z score is calculated for the conversation as $\frac{length-\bar{x}}{\sigma}$. Any conversation with a length of 2.5 standard deviations from the mean receives a length score of 0 to indicate recommendations that are too short/long. There are additionally differing penalty scales for conversations to the left of the mean which receive a 1.5 scale penalty, and conversations to the right of the mean which receive a 1.35 scale penalty. This scaling is meant to introduce lower scores for conversations that are too short and penalize conversations that are too long, but not as stiff a penalty as short conversations. The final score is normalized between 0 and 1.

The readability score is calculated using the Flesch-Kincaid reading ease test. [22] The number of words used in the recommendation is calculated, and the number of syllables in each word is calculated using the Python Pyphen package. The readability score for each conversation was then calculated per the test formula which is:

$$206.835 - 1.015 * (\frac{TotalWords}{TotalSentences}) - 84.6 * (\frac{TotalSyllables}{TotalWords})$$

Word importance scores are calculated based on TF-IDF values and rely on the TfidfVectorizer from the sklearn.feature_extraction.text library. Each recommendation is converted into a single string, which is then collated into a list where each element is the recommendation as a single string. The TF-IDF values are calculated for the corpus using the fit_transform() method, and then the sparse matrix is pruned to only hold nonzero values using the .nonzero() method into a matrix holding the TF-IDF score for each word in the corpus. The word importance score for the conversation is calculated as the summation of the TF-IDF scores of each word in the conversation by referencing the corresponding TF-IDF score for that word in the TF-IDF matrix.

The repetition score is calculated after removing stop words from the recommendation. Then the count of each word in the recommendation is checked. Any words with counts

greater than one add a point to the repetition score for that recommendation.

Subjectivity and Polarity are calculated using the TextBlob library where each recommendation is passed into the TextBlob object and the polarity and subjectivity member values are extracted from the object.

The grammar of a recommendation is calculated after removing stop words and uses the PyspellChecker library. Each word in the recommendation is run through the spellchecker, anything misspelled is recorded and the final grammar score is calculated as the total number of misspelled words over the length of the recommendation. In the E-redial dataset movies are placed in brackets [], and in INSPIRED mentioned movies are kept in a list. Any words in the movie brackets or within the INSPIRED list are not counted towards grammatical errors as movie title may contain abnormal spellings or nonsensical words.

Feature appearance is scored based on cosine similarity between the seeker and recommender's respective contributions. When collating the data to generate the feature appearance score, each speaker's messages are sent into separate arrays, where everything the seeker said is stored in a seeker array, and everything the recommender said is stored in a recommender array. This heavily disrupts the organization of the conversation but collects each speaker's conversational contributions individually. The speaker and recommendation arrays are then sent to BART for summary. The arrays are joined into one string, tokenized by the BART tokenizer, and the tokens are then passed into the BART model to generate an encoding of that particular speaker's contributions to the conversation. The summary IDs generated by BART are then decoded by the BART model, which returns a text with the main points of a speaker in the conversation. This summary generation is done for the speaker and the recommender. The pretrained bart-large-cnn tokenizer and model were used in this process. Once both speaker's conversation summaries have been generated, the summaries are embedded using BERT. The conversation summaries are tokenized with the BERT tokenizer, and the tokens are then passed through the BERT model, and the last hidden

state of the BERT model's output is taken as the embeddings for that conversation summary. When both summaries have been encoded using BERT, the cosine similarity between the seeker's embeddings and the recommender's embeddings is calculated, and recorded as the feature appearance score for that particular conversation. The cosine similarity between seeker and recommendation embedding summaries is a good heuristic for recommendation quality because it indicates that the recommender and seeker were focusing on similar items and a shared vocabulary. Cosine similarity is additionally scale invariant and effective for high dimensional data, which is helpful in this case as the recommender is almost always more verbose as they provide summaries of movie plots and review information so the recommender's summary embedding tends to be larger in scale and magnitude, being able to capture the similarity between the two summaries without having to adjust for the recommender's scale difference is convenient.

Each conversation is processed as above to extract it's quality indicator scores, and the results are recorded into a combined dataset of modified E-redial and INSPIRED conversations where the training data has 1557 conversations and the test data has 249 conversations. Figure 2 shows information about the score value of each quality factor.

After each conversation was processed and scored, target labels were assigned based on the values of each quality factor for a particular conversation. Three categories were created to classify conversations as either good, okay, or bad. The rules for determining class labels can be seen in Figure 3. For a conversation to be classified as good the score of each quality factor must meet every condition in that column. Some values need to be greater than or equal to a set value while others need to be less than or equal to a value, this is based on the meaning of the quality factor score. High grammar scores indicate many grammatical errors so grammar scores were set to be less than or equal to a particular threshold value, whereas feature appearance values must be greater than or equal to a threshold value as higher feature appearance scores mean better topicality for the conversation.

Training Data Quality Factor Statistics						
	Minimum	Maximum	Mean	Median	Range	Standard Deviation
Length	0.000	0.993	0.418	0.325	0.993	0.233
Readability	-50.871	110.198	74.149	78.427	161.070	18.856
Word Importance	3.808	13.611	8.814	8.780	9.804	1.534
Repetition	1.000	80.000	16.475	15.000	79.000	9.527
Subjectivity	0.189	0.821	0.554	0.555	0.632	0.088
Polarity	-0.105	0.775	0.290	0.289	0.880	0.123
Grammar	0.000	0.130	0.018	0.015	0.130	0.014
Feature Appearance	0.597	0.946	0.838	0.849	0.349	0.055
Test Data Quality Factor Statistics						
Length	0.000	0.992	0.424	0.325	0.992	0.240
Readability	-4.042	110.198	83.428	87.185	114.240	14.406
Word Importance	3.771	13.665	8.517	8.409	9.894	1.554
Repetition	1.000	59.000	14.580	13.000	58.000	8.454
Subjectivity	0.189	0.821	0.559	0.561	0.632	0.088
Polarity	-0.105	0.775	0.310	0.309	0.880	0.123
Grammar	0.000	0.130	0.017	0.014	0.130	0.015
Feature Appearance	0.613	0.946	0.844	0.854	0.333	0.052

Figure 2: A table showing various statistical values for each quality indicator's score, where the training data breakdown is on top and the test data breakdown is in the lower half of the chart.

After scoring each conversation the final distribution of labels may be seen in Figure 4. Good conversations comprise roughly 20% of the labels across both the training and test dataset. However in the training dataset okay conversations make up the plurality at 44% of conversations, and in the test dataset, bad conversations make up the plurality of conversations at 46% of the dataset. This is a quirk of the E-redial dataset [8] where the authors noted that some conversations 823 of the system responses in the test set are idle with no movie recommendations, meaning the majority of conversations imported from the E-redial test set are just chit-chat and thus not good recommendation dialogue.

Quality Factor:	Target Label Requirements		
	Good (0)	Okay (1)	Bad (2)
Length	>= 0.3	>= 0.25	Default case. If a conversation did not meet all the required scores to be classified as good or okay, the conversation would be assigned a bad label (2).
Readability	>= 0.7	>= 0.5	
Word Importance	>= 7	>= 4	
Repetition	<= 20	<= 25	
Subjectivity	>= 0.45	>= 0.3	
Polarity	>= .15	>= .1	
Grammar	<= 0.04	<= 0.05	
Feature Appearance	>=.82	>=.8	

Figure 3: A table depicting the rules for label assignment. Each quality factor lies on the left, and each quality label occupies a column. So long as a conversation met all scoring requirements it could be assigned the label of the corresponding column.

Network Design

There is a single network design, with slight variations are made to accommodate three separate base models. The GPT2, NEO, and t5 models are used as base models for their powerful capabilities. There is an encoder network that passes a conversation through the base model to get the encodings and attention masks of that conversation, as well as converting each quality factor to a tensor of appropriate size and shape.

Once the data has been embedded and converted to tensors it gets passed to the classifier network. In the classifier network, the weights of the base models are frozen for efficiency, and because in initial testing it was observed that weight adjustments to the base models often resulted in negative transfer learning. The classifier network is comprised of three blocks with three linear components in each layer. In each layer, there is a residual connection between the first and final component, and each layer component shrinks the hidden layer size gradually. Between layers the hidden dimension size is increased such that each layer starts wide, and then shrinks down to a small size. Each model uses cross entropy as the loss function, relu as the activation function, and a multiclass accuracy metric. A diagram of the architecture may be seen in Figure 5 the input layers, and Figure 6 the residual block layers.

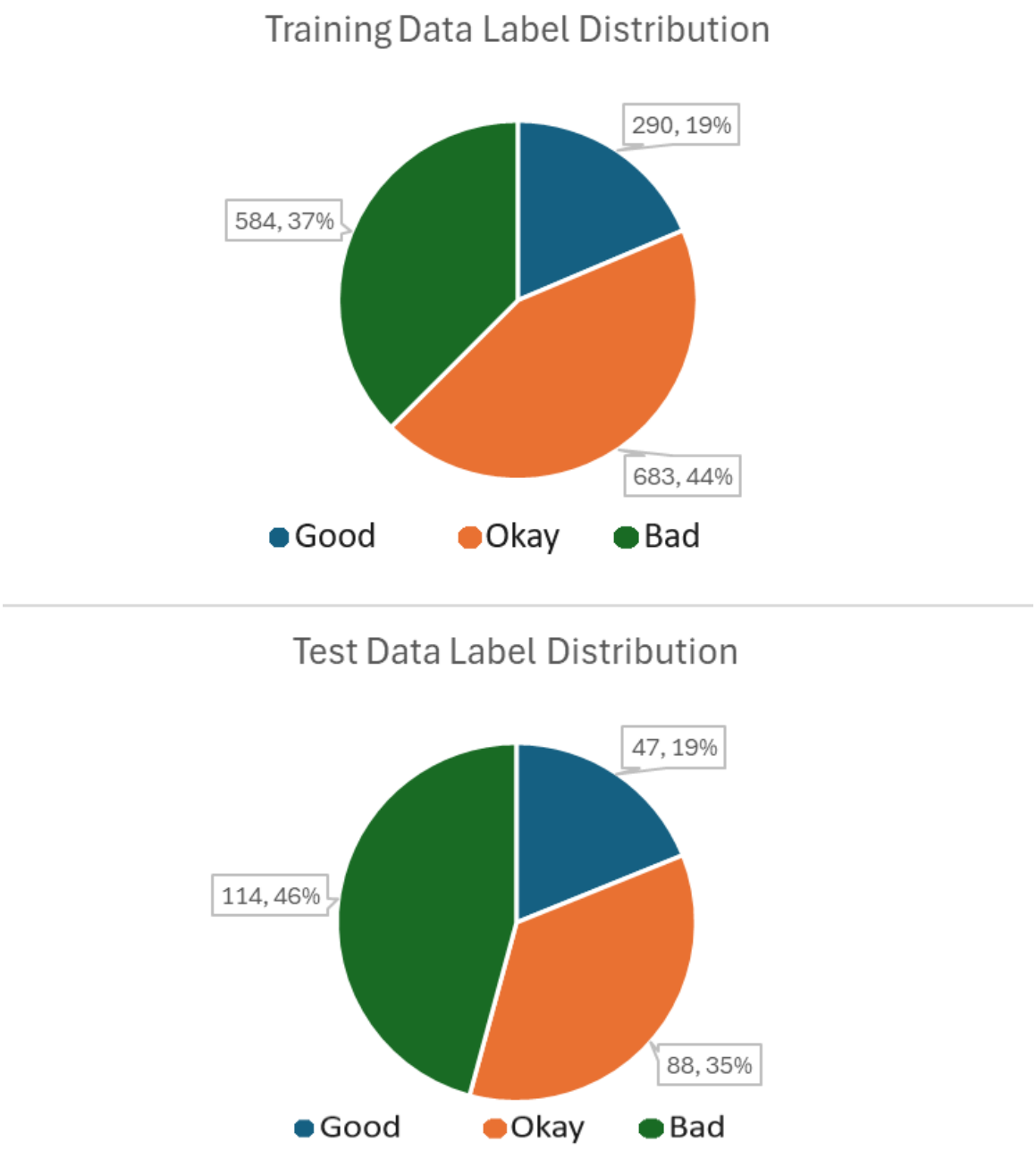

Figure 4: A chart depicting class label distribution after target labels were generated for both training (top) and test data (bottom).

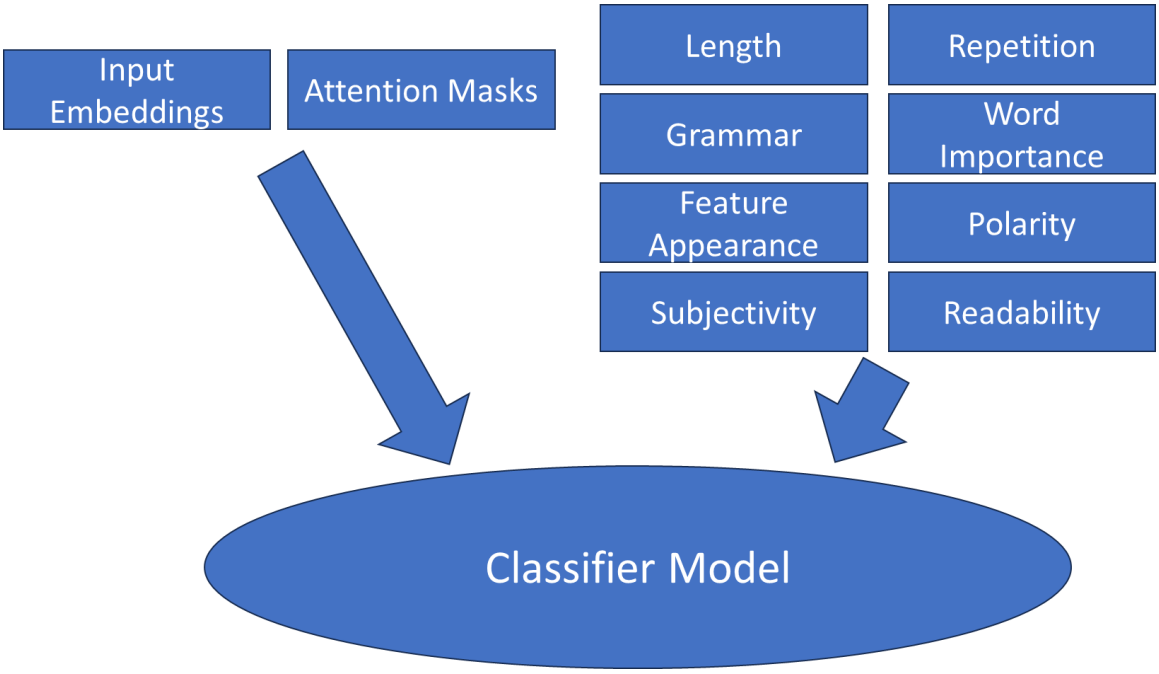

Figure 5: A flowchart showing the input architecture of each model, without the base model. The architecture pattern is the same across each of the three models, with only the base layer (GPT2,NEO,t5) changing. The model takes in the final output of the embedded conversation and the attention masks, as well as each of the eight quality score indicators, which are then combined together to be processed through blocks that can be seen in Figure 6. The architecture features 3 repeated blocks of shrinking linear layers with a residual connection between the first and last linear.

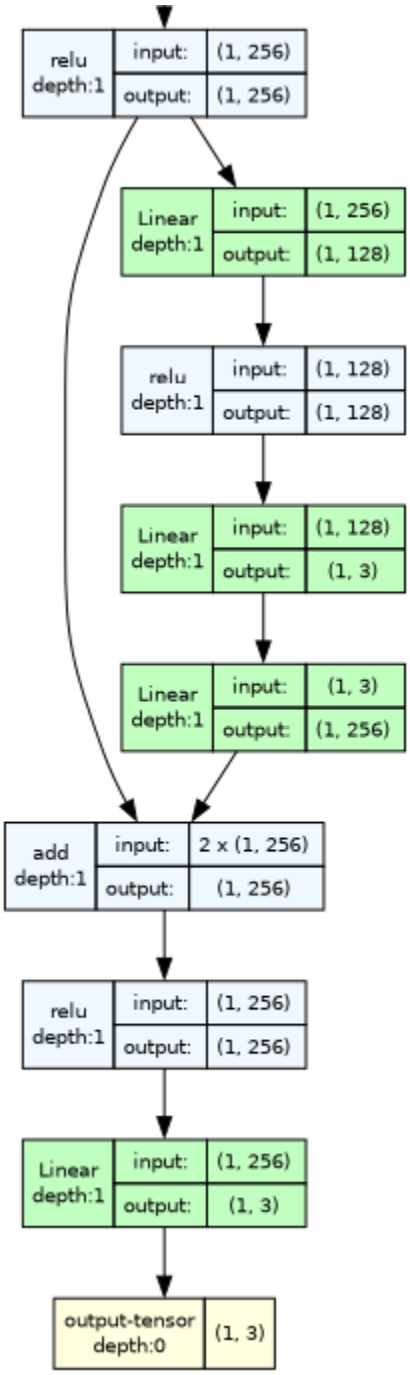

Figure 6: A flowchart showing the final block and output layer of the model. Each model features 3 repeated blocks of shrinking linear layers with a residual connection between the first and last linear.

In the forward pass, the conversation is embedded using the base model, and the mean of the last hidden state is used to represent the conversation in the embedding space. The quality scores for the conversation are repeated to match the shape of the embedded conversation, and then the scores and the conversation are concatenated together. Once the conversation and scores have been combined they are passed through the layers of the model.

Testing Procedure

The base model networks GPT2, NEO, and t5 are tested over 30 independent runs for 80 epochs. Due to abnormal sizes of the training and test datasets, the batch size for the training dataset is 9 and the batch size for the test dataset is 3. This testing procedure is used for nonstandardized data and is repeated with the same parameters, but substitutes standardized data for each quality score. Each model's loss and accuracy for training and validation epochs are recorded. Once each model had been trained the best version of each normalized model was selected for ICE and SHAP analysis, and the results were recorded.

CHAPTER IV.

Results

The validation results of the nonstandardized models is consistent across all three model types with the average results being quite similar. The accuracy and loss results of the nonstandardized models may be seen in Figure 7. The loss percentages are located at the top of Figure 7, and the accuracy percentages are located at the bottom of Figure 7. It can be observed that the accuracy rating for each of the three models remains near 50%, which is roughly at change percentage for correctly identifying a conversation as good, okay, or bad. The loss for all three models ends around .85. The accuracy and loss lines never converge or come close to crossing. In this experiment, the nonstandardized models perform the worst.

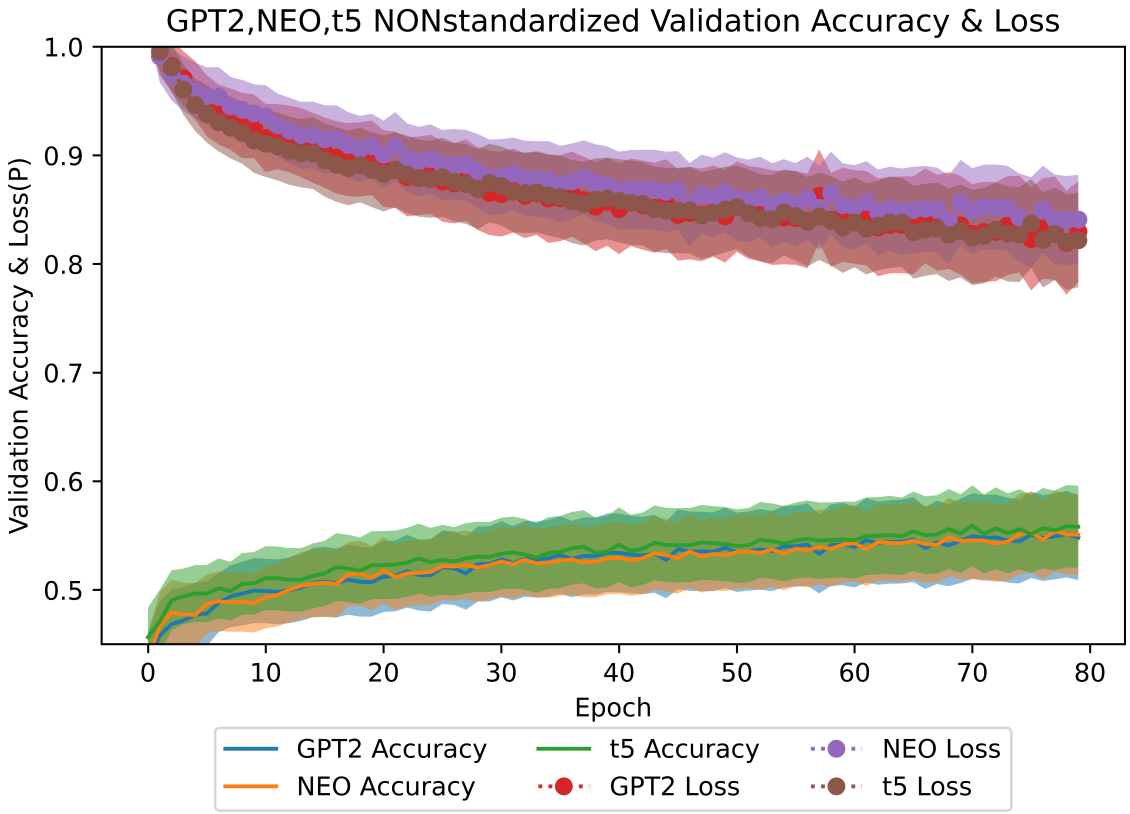

Figure 7: An image showing the validation loss (top, and in dotted lines) and accuracy (bottom, and in solid lines) ratings for each of the three models, GPT2, NEO, and t5 using nonstandardized quality factors with standard error regions representing a 95% confidence interval.

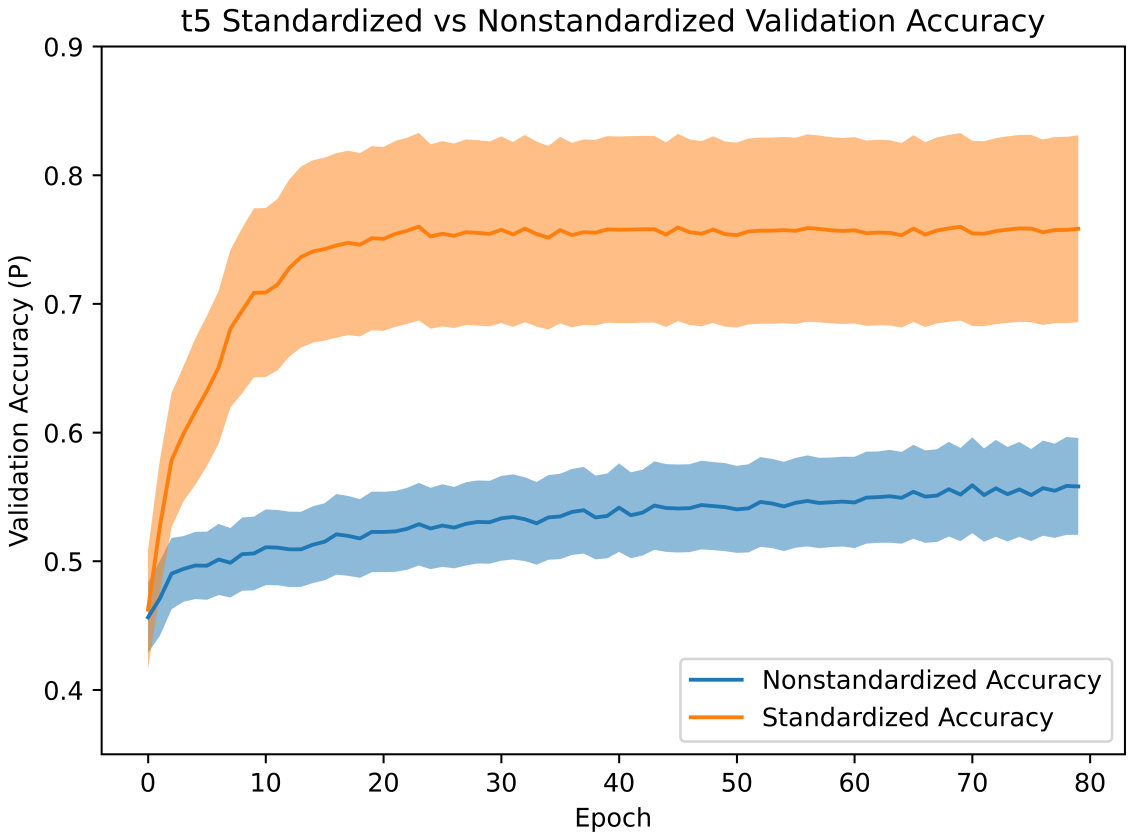

Figure 8: An image showing the standardized (top) and NONstandardized (bottom) validation accuracy of the t5 model with standard error regions representing a 95% confidence interval.

The standardized and non-standardized validation accuracies of the GPT2, NEO, and t5 models may be seen in Figures 8,9, and 10. Across all three base models the general trend is clear that standardization of quality factors has a strong positive effect on model accuracy. For each of the three models, the nonstandardized versions report average accuracies of 50%. The standardized models report above 70% accuracy for all models. The t5 model (Figure 8) performs the worst at an average accuracy of 75%, GPT2 (Figure 10) achieved an average accuracy 5% higher than t5 with an average accuracy rating of 80%, and the NEO model (Figure 9) performed best overall with an average accuracy around 83%. In Figure 8, the t5 model shows the broadest variance in performance, but overall, each model demonstrates similar behavior as all Figures 8, 9, and 10 have similar shapes and values. The similarity in

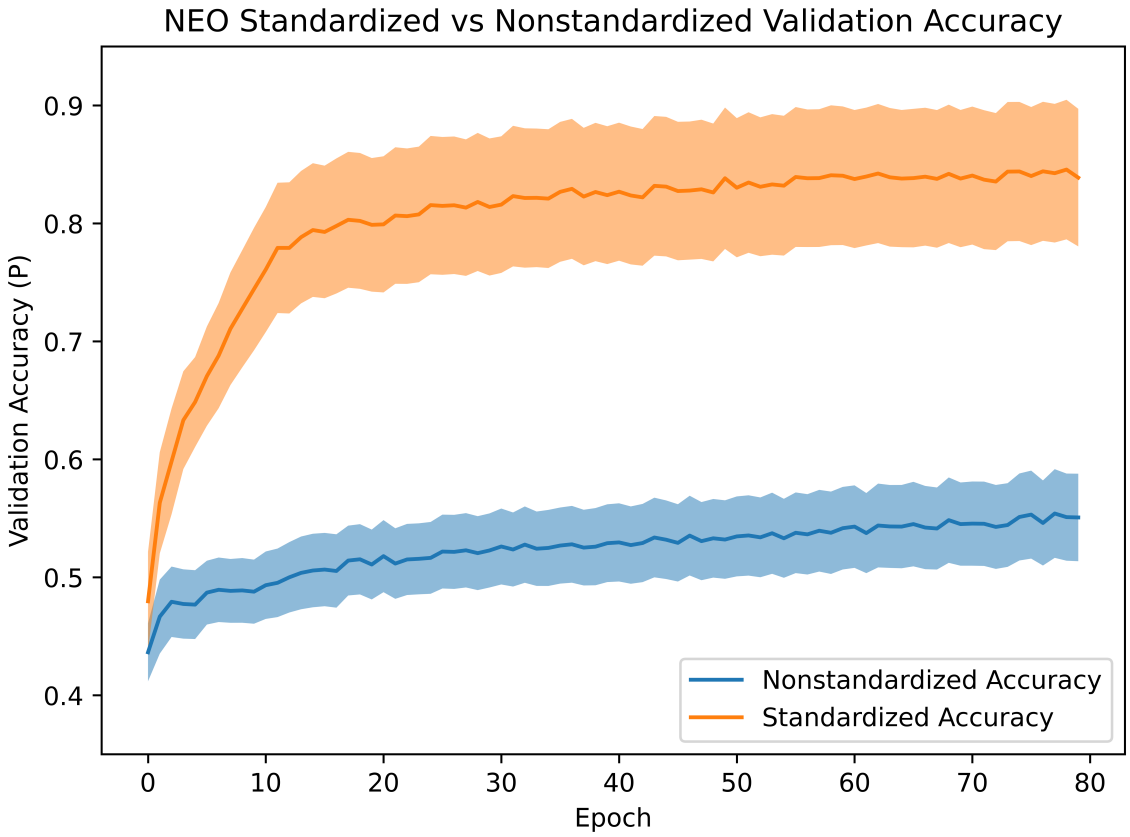

Figure 9: An image showing the standardized (top) and NONstandardized (bottom) validation accuracy of the NEO model with standard error regions representing a 95% confidence interval.

performance can be seen in Figure 11, which provides a summary of each model's average classification accuracy. The accuracy for each of the three models in 11 improves rapidly in the 0-15 epoch range, and then plateaus or only grows slightly in the 16-80 epoch range. Seen this way, it is clear that NEO is the best-performing model, and t5 the worst.

Each of the models performs better on the training set which can be seen in the right side box plots of Figures 12, 13, and 14. Standardized models perform better than nonstandardized, and performance is higher in all Figures for training data. The median performance of each model in Figures 12, 13, and 14 shows higher accuracy ratings than the average accuracy ratings shown in Figures 8,9, and 10, with the median accuracy ratings for t5 in

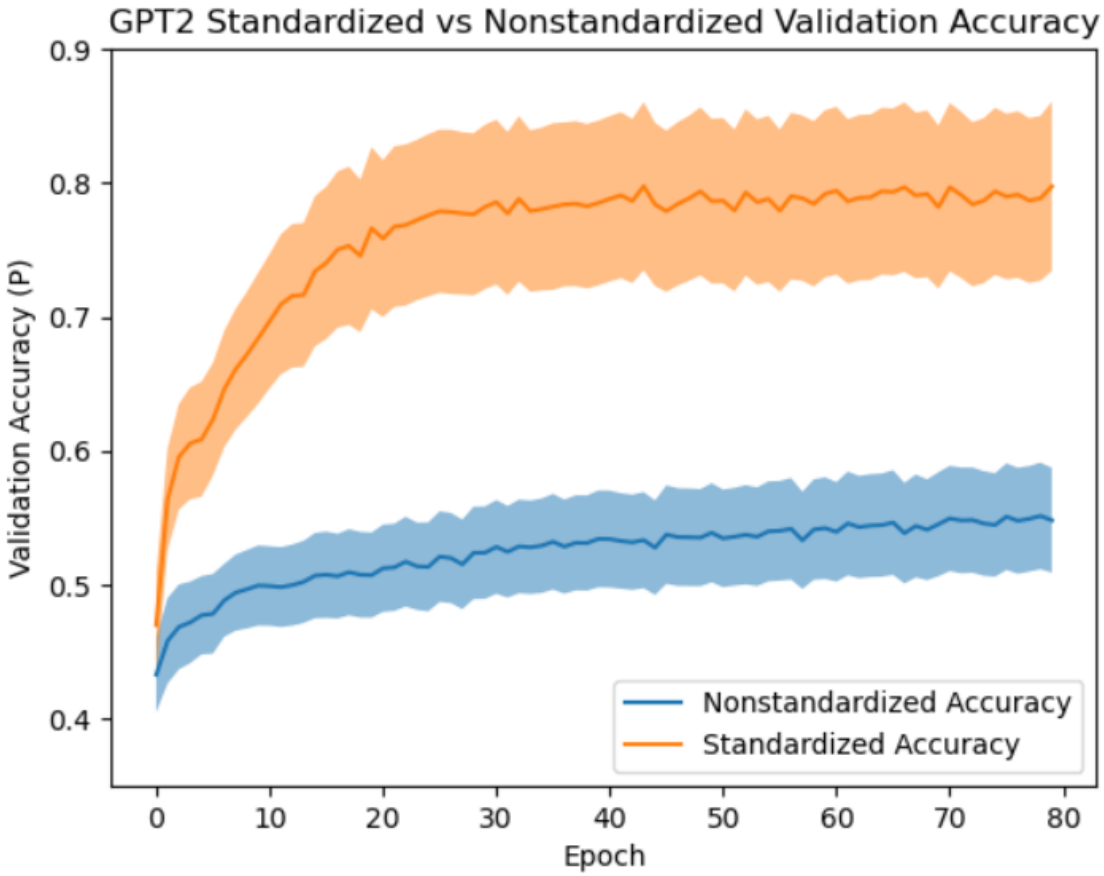

Figure 10: An image showing the standardized (top) and NONstandardized (bottom) validation accuracy of the GPT2 model with standard error regions representing a 95% confidence interval.

Figure 14 looking fairly competitive to GPT2 in Figure 12. NEO, in Figure 13, remains the best-performing model overall. NEO performing the best can be most easily seen in Figure 15. In Figure 15, the validation accuracy for GPT2 and t5 have overlapping notches, whereas NEO performs best with an approximate median validation accuracy of 86%.

The results of the ICE analysis may be seen in Figures 16, 17, and 18. The ICE analysis was done on the test data with standardized quality factor scores, where for each conversation in the test set the model predicted the class of the conversation where all data was held constant except for the value of a quality factor. The quality factor's value varied from

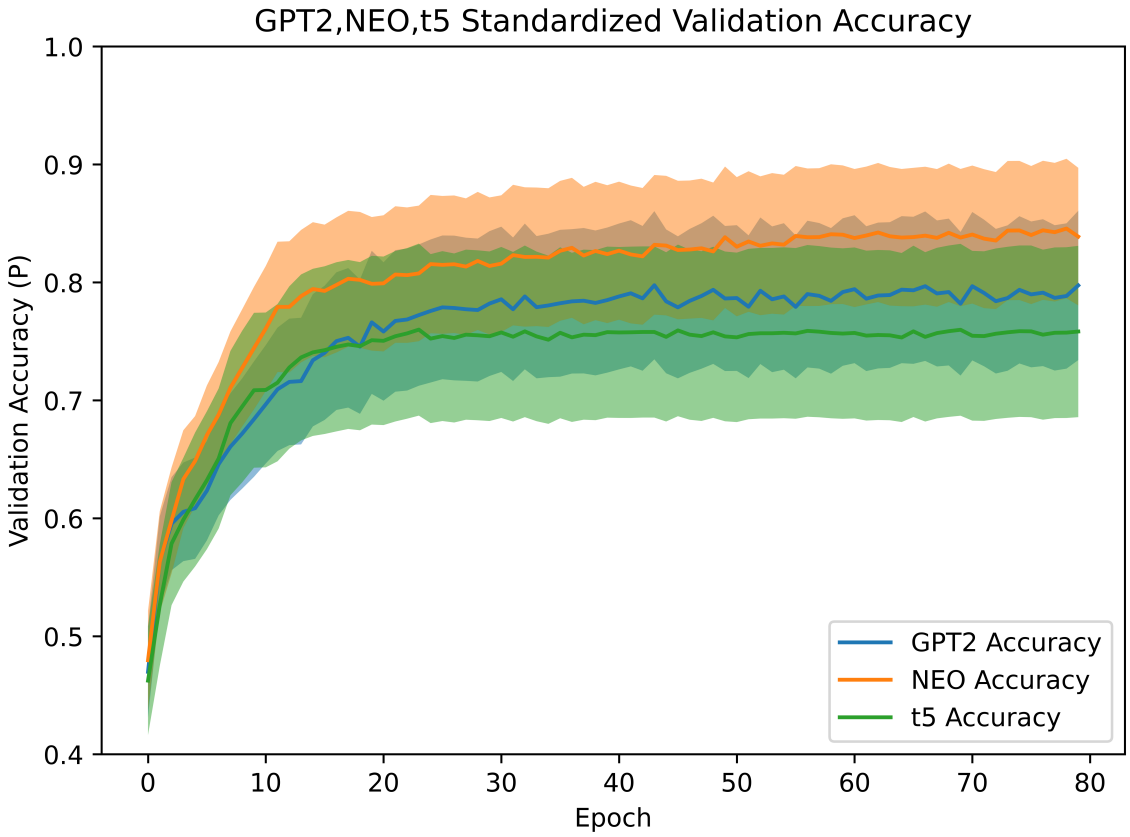

Figure 11: A chart displaying the standardized validation accuracy of the GPT2 (blue), NEO (orange), and t5 (green) models with 95% confidence intervals shown.

[0.0-1.0] in increments of .01 and the average and median values of the classification were recorded for each quality factor on every conversation in the test set. For all models, the mean (blue) and median (green) results of the ICE analysis on length, readability, word importance, polarity, and subjectivity quality factors result in straight or very slightly curved lines. Regardless of the variations in scores for length, readability, and word importance, all three models show very strong tendencies to classify the conversation as okay (1).

The values of the lines differ particularly for the polarity and subjectivity quality factors where the GPT2 and t5 models report changes to the subjectivity and polarity values on average push the model towards okay (1) or bad (2) classifications. The NEO model also

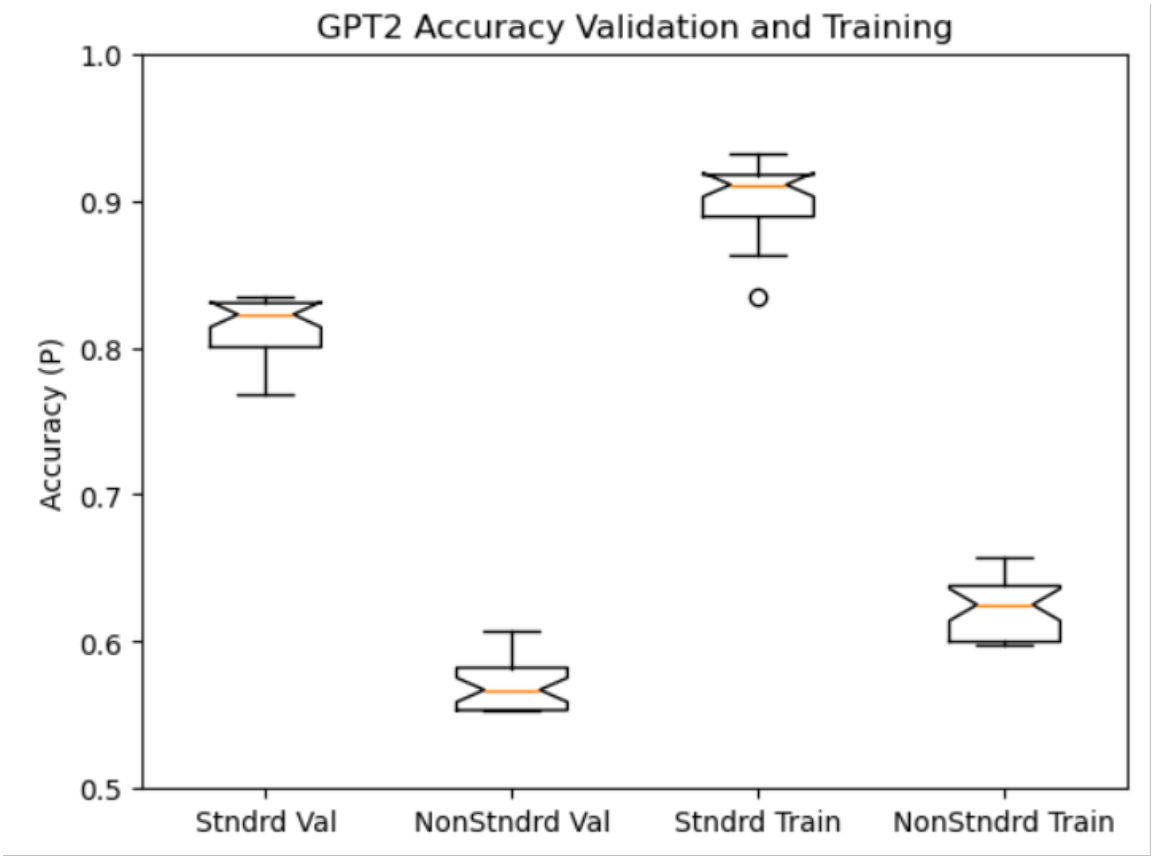

Figure 12: An boxplot showing the validation (left) and training (right) accuracy, across standardized and NONstandardized versions, of the GPT2 model.

shows this tendency but reports much more strongly with a higher average class value and a median class value of bad (2), whereas the median for GPT2 and t5 is the okay (1) class.

The results for the quality factors in the bottom right of each ICE chart, repetition, feature appearance, and grammar, show more variance than the other five quality factors. The average value for repetition in all models shows that lower repetition values push the models towards good (0) classification and the median value shows this tendency as well. As the repetition value grows past the .5 each model begins to classify conversations as bad more frequently. The average ICE repetition value for GPT2 and t5 is roughly equal in magnitude, where lower values push for good (1) classification, and higher value push for

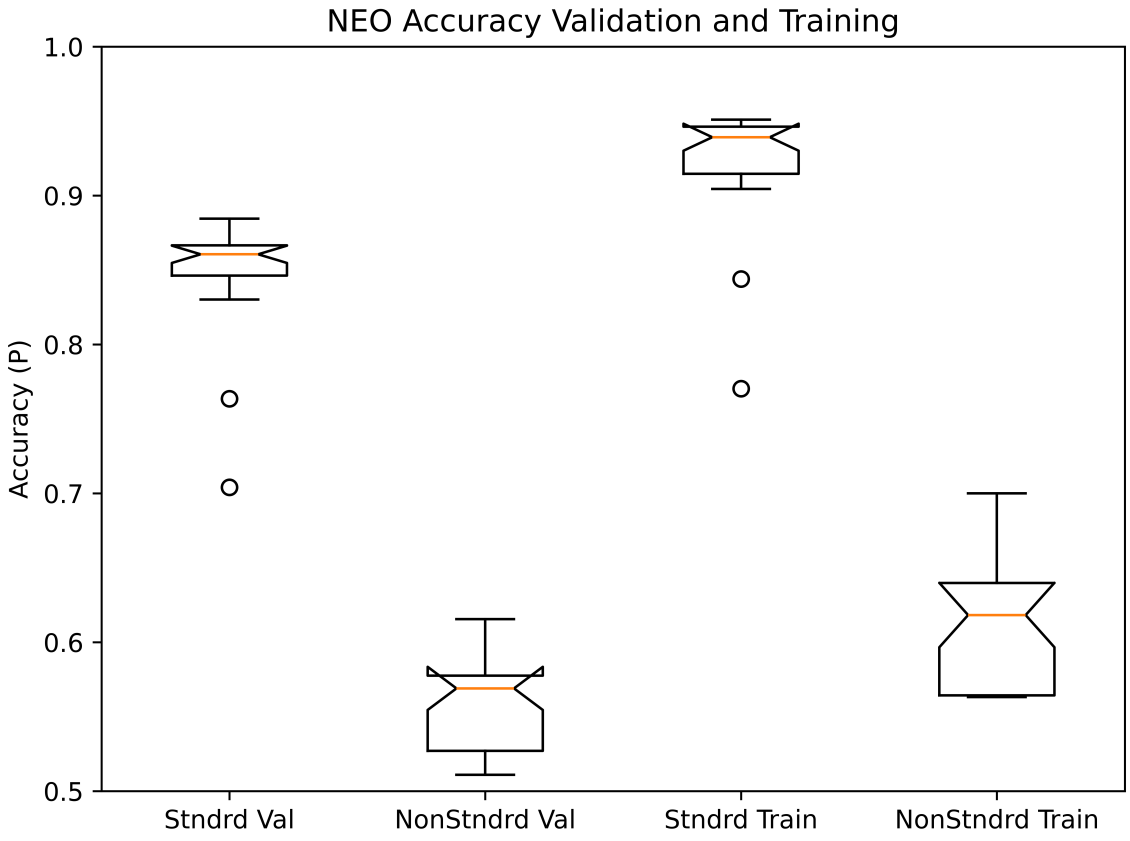

Figure 13: An boxplot showing the validation (left) and training (right) accuracy, across standardized and NONstandardized versions, of the NEO model.

bad (2) evaluations. The average ICE repetition value for NEO is perhaps more interesting, where compared to GPT2 and t5 higher repetition values push the average classification much more strongly towards the bad (2) class.

For the feature appearance score, each model on average classifies a conversation as bad (2) as the value of feature appearance grows past the .5 value. The median classification jumps to the bad (2) class in the NEO model once the feature appearance value grows past .8, and the average classification for the t5 model takes on strong linear growth towards bad (2) classification once the feature appearance value grows beyond .6. . This goes against the hypothesis that the more a recommendation shares common features with a request the

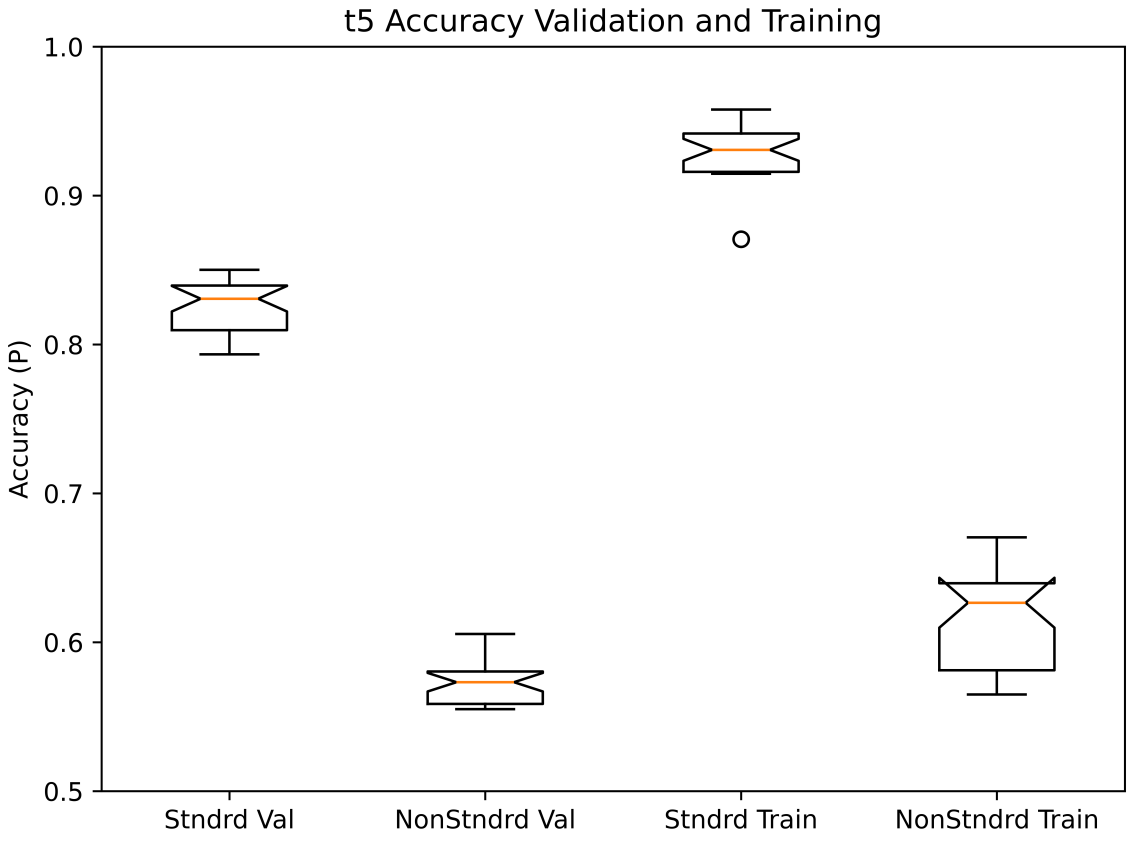

Figure 14: An boxplot showing the validation (left) and training (right) accuracy, across standardized and NONstandardized versions, of the t5 model.

better the recommendation would be, and indicates all three models missed the mark on this particular quality factor.

The grammar quality factor for the t5 and NEO models show that variations in grammar have strong tendencies to push towards a bad (2) classification, where the median values for both models always predicts bad (2), and as the value of the grammar feature increases the average prediction trends towards 2. This is not the case for the GPT2 model, where the median value is always the okay (1) class, however, the GPT2 model on average trends towards a bad (2) classification as the value of the grammar score grows larger.

The results of the SHAP analysis may be seen in Figures 19, 20, and 21. The SHAP

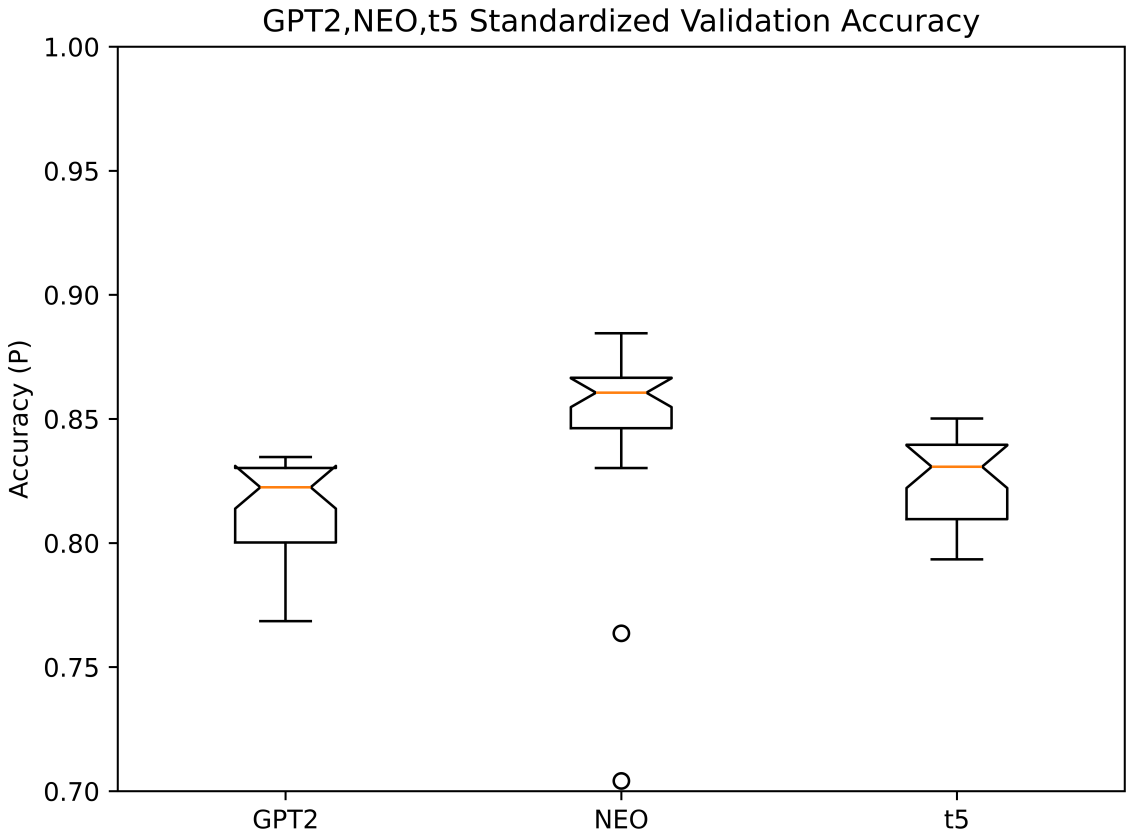

Figure 15: An boxplot showing the standardized validation accuracies of the GPT2 (left), NEO (middle), and t5 (right) models, with notches displaying a 95% confidence interval.

analysis is done by establishing a background dataset, the first 83 conversations in the test data. For every conversation in the dataset, and every quality factor in a conversation, the value of the quality factor is replaced with a random value of that same quality factor in the background dataset. The model will then predict the class of the conversation with the adjusted data. The SHAP value is the altered prediction label minus the original prediction based on the unmodified data. This means the max SHAP value is 2 where the model predicted a label of 2, but the original value was class 0, and similarly the floor is a SHAP value of -2, where the model predicted class 0 with the altered data, but the original prediction was class 2. The Y axis in Figures 19, 20, and 21 thus represents the SHAP value, and not a

class label. Correspondingly, positive values indicate the adjustments to the data pushed the model to predict either the okay (1) or the bad (2) class and negative predictions indicate the changes pushed the model to predict the okay (1) or good (0) class. SHAP values of 0 indicate the changed data had no practical effect as the model still predicted the same class even with adjusted data values. The wider the spread of dots on the Y axis indicates how frequently the model predicted a particular SHAP value across the test data.

All models exhibit similar tendencies, where the most common occurrence is for the alternative data to have a neutral impact on class prediction as each plot shows the thickest and widest clustering around SHAP value 0 for each quality factor. This indicates individual factors are not massively impact independent of the other factors. Similar to the ICE analysis, the factors that show the most impact on model predictions are repetition, feature appearance, grammar, and polarity also shows the ability to push the model to predict class 0 or 2.

Each quality factor in Figures 19, 20, and 21 demonstrate the ability to push the model towards any of the three classes, with no quality factor having entirely neutral effects regardless of alterations to its value. Of the eight quality factors, the SHAP analysis shows that readability and subjectivity have the weakest effects where all models in Figures 19, 20, and 21 show readability and subjectivity factors having strong neutral tendencies and only modest capabilities to push the model towards the bad or good classes. Of the three models, GPT2, in Figure 19, demonstrates the highest reactivity towards variations in quality factors as it is the only model where alterations to repetition, polarity, feature appearance, and grammar scores can push the model to predict any class. As with the accuracy and ICE analysis, each model demonstrates consistent behavior similar to the other models.

The validation results for nonstandardized models across all three types (GPT2, NEO, and t5) consistently show similar average performance. Figure 7 depicts the accuracy and loss percentages, indicating that all models maintain around 50% accuracy. Nonstandardized models exhibit the lowest performance in this experiment. Standardization of quality

factors notably improves model accuracy, as demonstrated in Figures 8, 9, and 10. The average accuracies for nonstandardized models hover at 50%, while standardized models achieve above 70% accuracy. NEO outperforms both GPT2 and t5, reaching an average accuracy of approximately 83%. Analysis of Individual Conditional Expectation (ICE) charts (Figures 16, 17, and 18) reveals that variations in quality factors affect model predictions differently. While factors like length, readability, and word importance show consistent trends toward classifying conversations as okay, others like polarity and subjectivity push models towards okay or bad classifications. Further, SHAP analysis (Figures 19, 20, and 21) illustrates how alterations in quality factors impact model predictions. Most factors exhibit neutral effects, with repetition, feature appearance, grammar, and polarity showing the most influence on predictions. In conclusion, standardizing quality factors significantly enhances model accuracy, with NEO exhibiting the best performance. However, variations in certain quality factors can still influence model predictions, highlighting the importance of careful consideration and fine-tuning in CRS development.

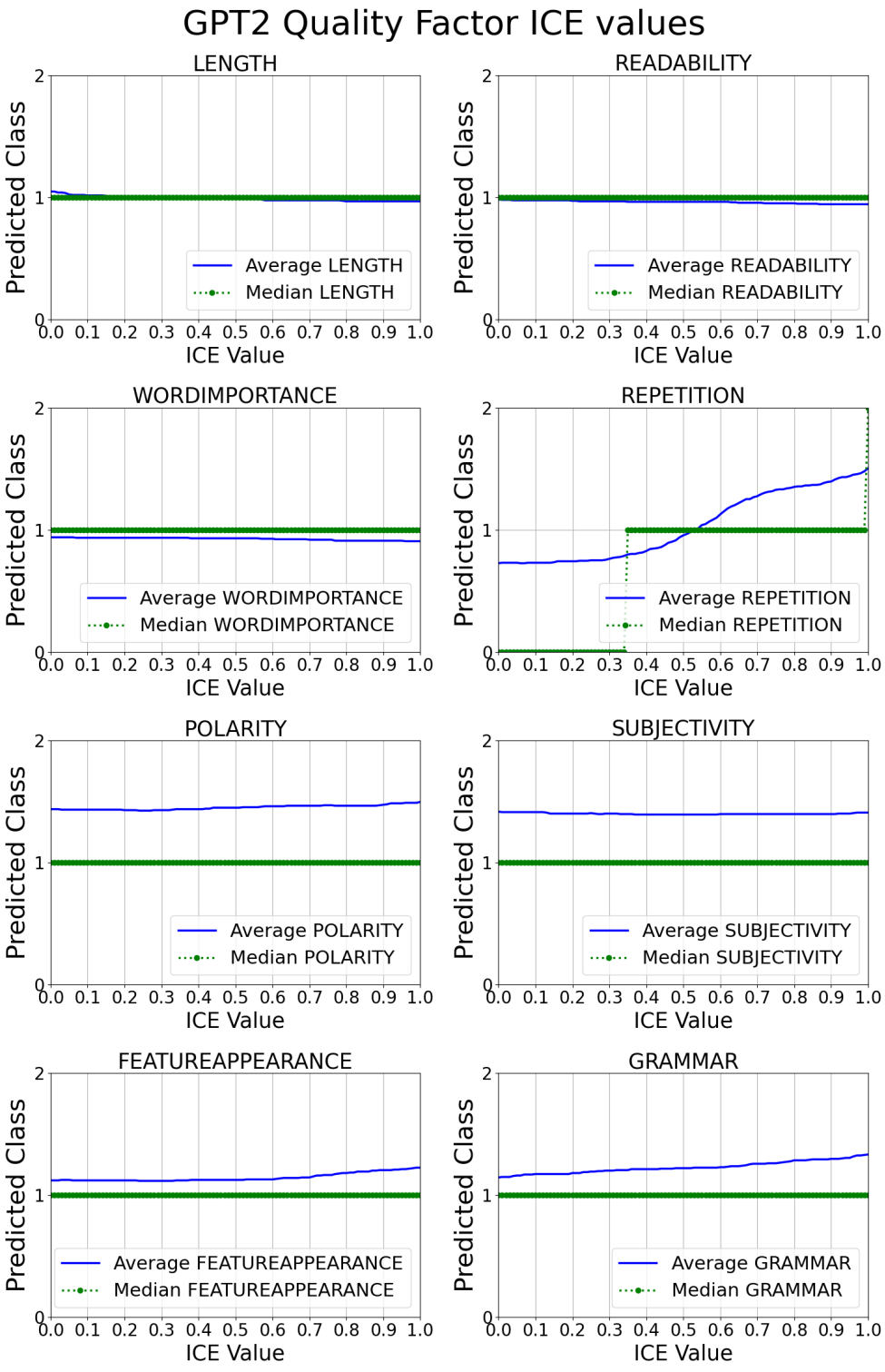

Figure 16: Figures displaying the average and median ICE values for the GPT2 model across eight quality factors, where the average is in blue and the median is plotted in green.

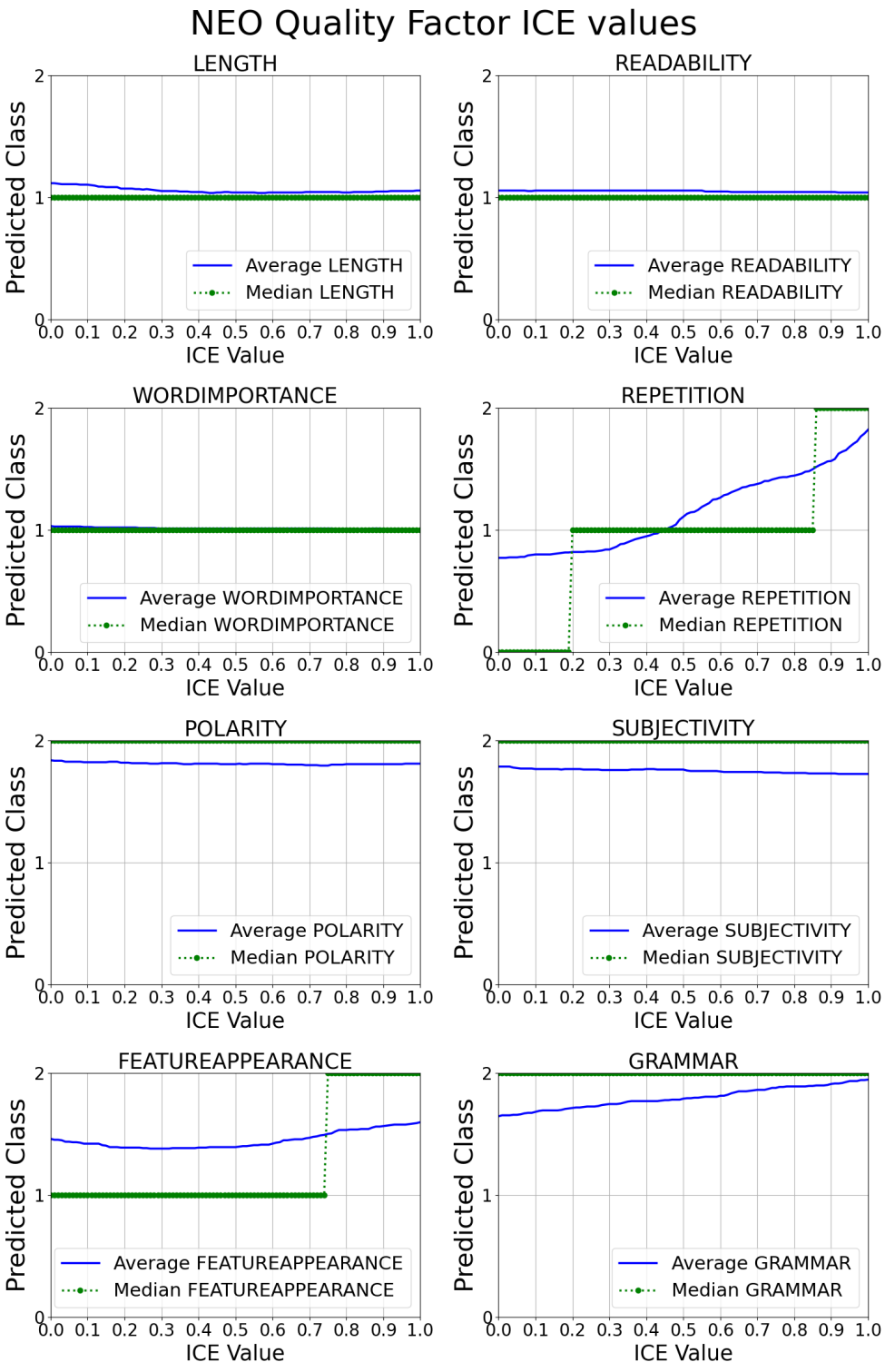

Figure 17: Figures displaying the average and median ICE values for the NEO model across eight quality factors, where the average is in blue and the median is plotted in green.

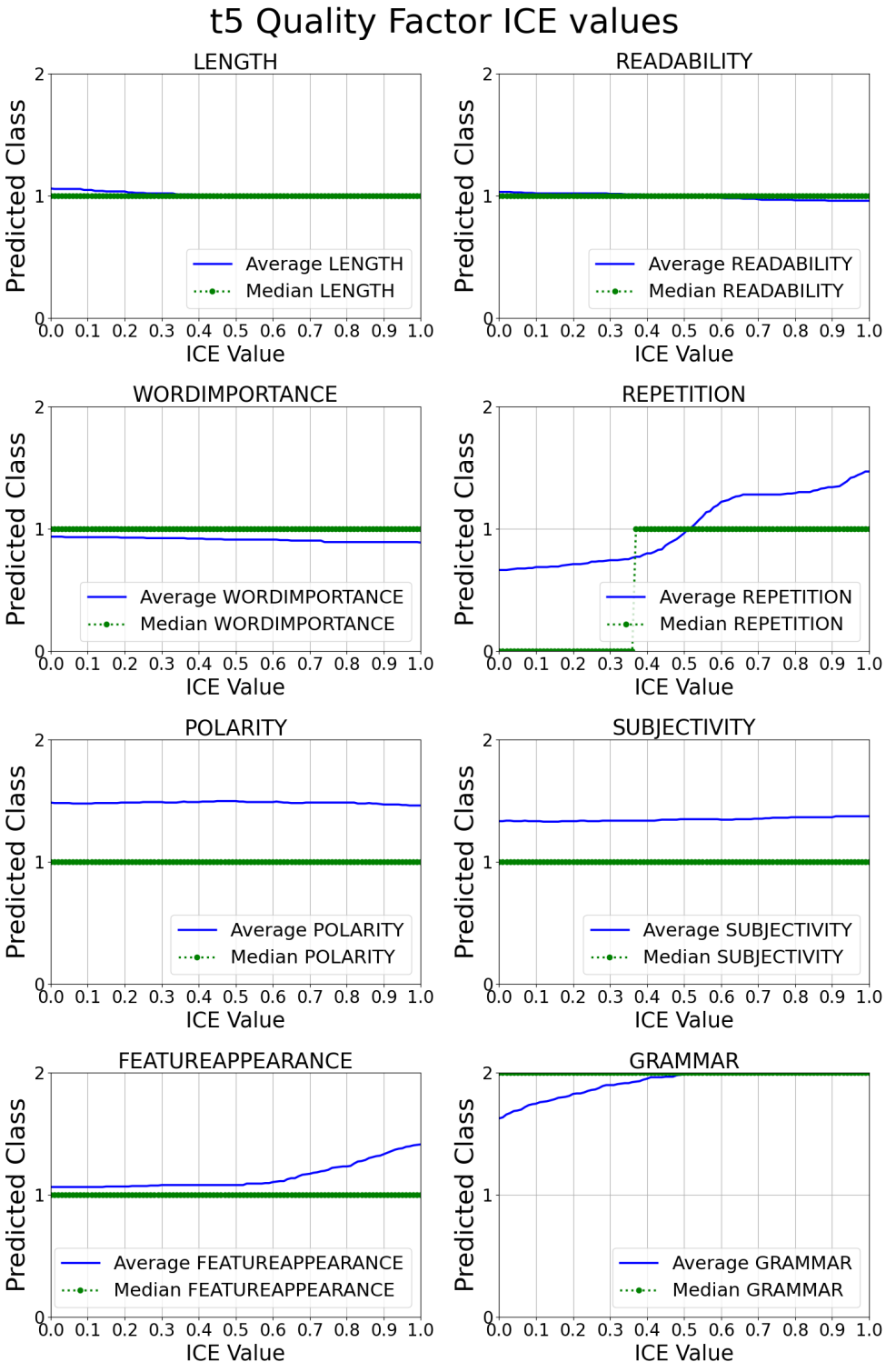

Figure 18: Figures displaying the average and median ICE values for the t5 model across eight quality factors, where the average is in blue and the median is plotted in green.

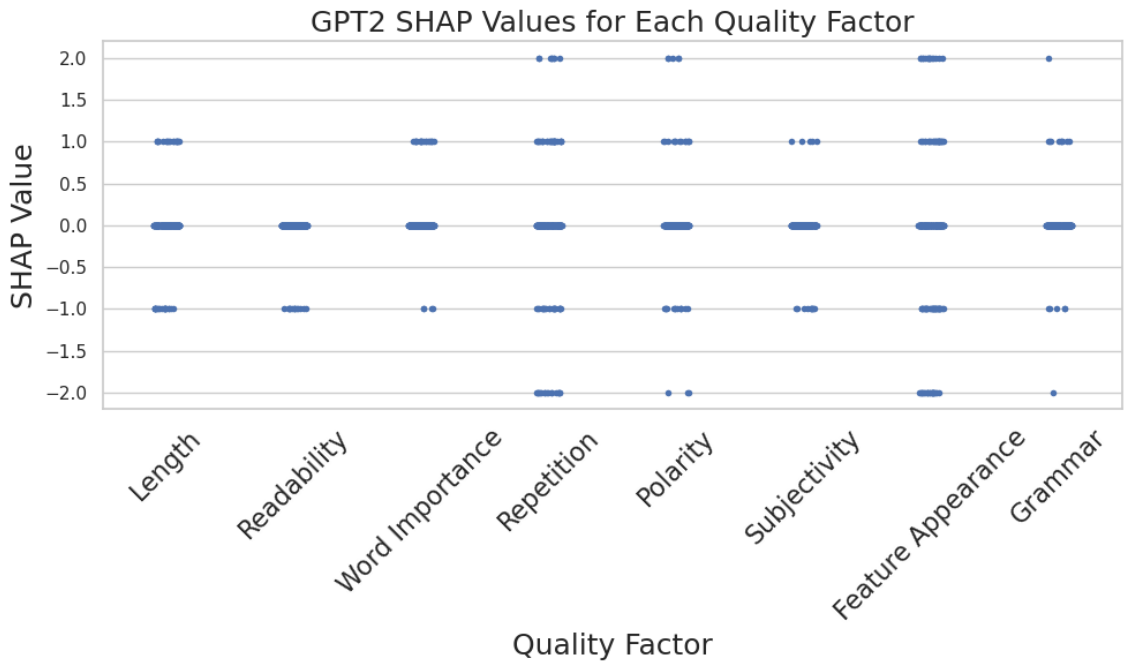

Figure 19: A beeswarm plot displaying the SHAP values for the eight quality factors in the GPT2 model.

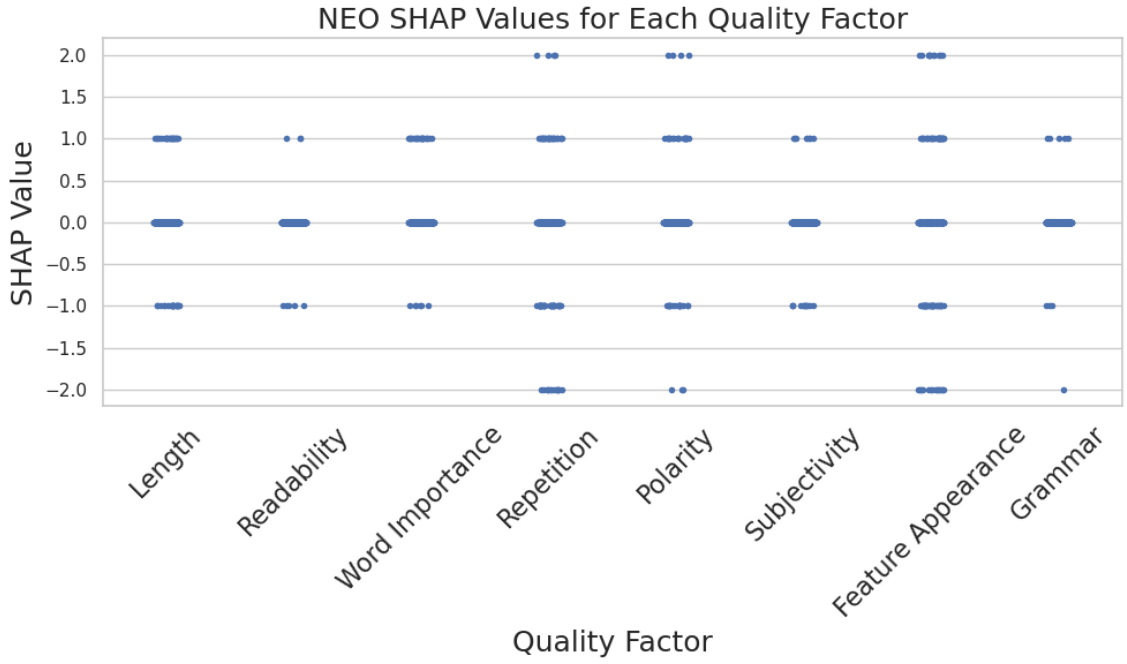

Figure 20: A beeswarm plot displaying the SHAP values for the eight quality factors in the NEO model.

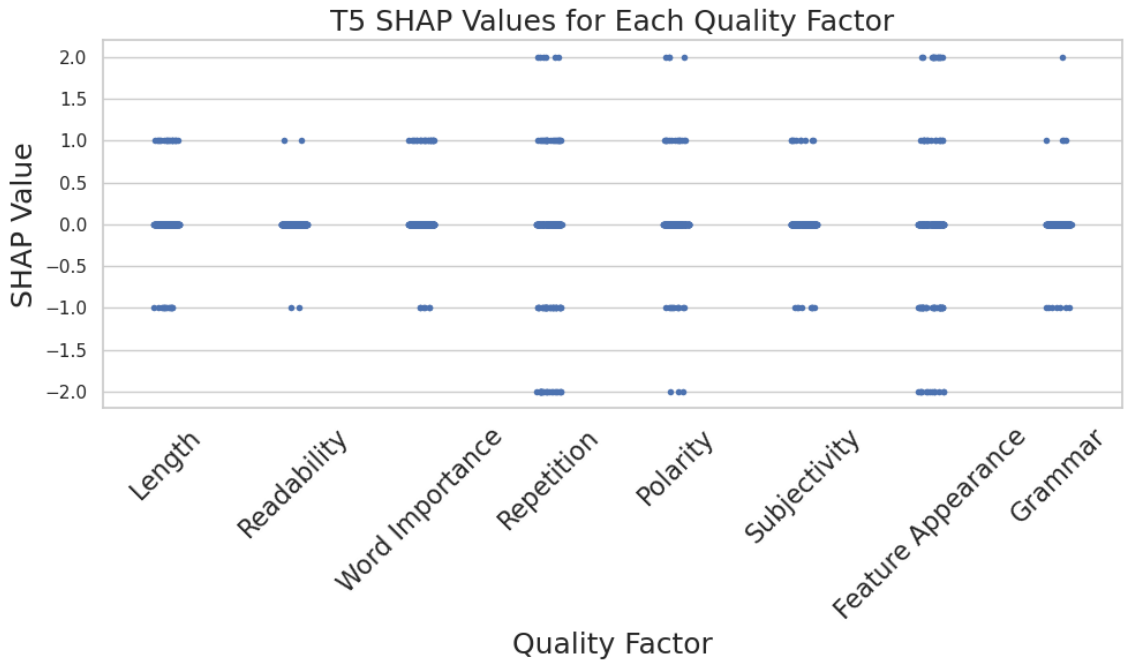

Figure 21: A beeswarm plot displaying the SHAP values for the eight quality factors in the t5 model.

CHAPTER V.

Discussion

Based on the results above in Figure 11, NEO is the best-performing model in terms of accuracy, it achieved higher accuracy scores than either GPT2 or t5 and performed better to a statistically significant degree based on Figure 15. The GPT2 model was a close second and both NEO and GPT2 had extremely similar behaviors across performance, and the ICE and SHAP analyses. This is unsurprising as NEO is an open source version of GPT2, and is designed to mimic the abilities of GPT2. It is believed that NEO was able to outperform the GPT2 model due to slight architectural differences. Both models have a window size of 256 tokens, but unlike GPT2, NEO uses local attention in every other layer. GPT2 and NEO were also trained on different datasets, and it is possible the extra diversity in the NEO dataset allowed NEO to model the conversational data better than GPT2.

The t5 model performed worst overall in terms of average prediction accuracy, but in Figure 15, t5 performs better than GPT2 when looking at median performance. This is most likely because the t5 model had more outliers than the other two models, which pulled the average accuracy of the model lower. The GPT2 and NEO models have auto-regressive objectives, where the models generate text tokens based on the previous token inputs, and contain decoder transformer architectures with attention mechanisms. Comparatively, the t5 model has a text-to-text objective and encoder and decoder transformer layers. The differences in architecture and training objectives between t5 and the other two models is likely the reason t5 had more instances where the model never converged. However, it is also likely t5's architecture that allowed it to have higher median performance than GPT2, meaning with extra fine tuning t5 could be a more consistent performer than GPT2.

Figures 12, 13, and 14 demonstrate two trends in performance. Firstly, each model performs markedly better on the training data than the test data. This is believed to be the result of particularly difficult conversations in the test dataset from E-redial. As mentioned in

Guo et al. [8], 823 system responses in the test set are idle with no movie recommendations made, as the workers creating the dataset did not follow instructions and requirements for the conversation. Both INSPIRED and E-redial were created with similar conversational requirements, and the model should have picked up on the pattern of conversations, and the fact that a large portion in the test set does not conform to the established pattern makes those conversations more difficult to classify. This can be fixed by diversifying the data the model is trained on, both by including recommendation conversations not centered around movies, and by including recommendations that have more varied lengths. The second trend Figures 12, 13, and 14 show is that standardization has a massive positive effect on classification accuracy across all 3 base models.

Standardization of quality factor scores likely has such a positive effect because it gives each score an easier-to-understand context. For instance, the repetition score is additive and goes up for every repeated word in a conversation, so repetition scores can become very high whole numbers. Grammar is scored similarly. However, polarity and subjectivity scores are always decimal numbers in the range [0.0-1.0]. Readability scores range from 0-100 based on the Flesh-Kincaid test, and so on. The scale of each quality factor differs from one another and the context is different as well. Higher repetition scores are meant to be interpreted as bad, but higher feature appearance scores is meant to be interpreted as good. There are many complex relationships to learn for how each quality factor score relates to the classification of the conversation. Standardization helps prevent larger-scaled features like repetition and readability from having an outsized impact on the model, and lets each quality factor contribute more equally to the learning process.

The ICE analysis charts (Figures 16, 17, 18) and the SHAP analysis charts (Figures 19, 20,21) demonstrate that individual variations in quality factor values show little ability to adjust model predictions. Combined, both the ICE and SHAP analyses appear to indicate that the models predict the class label based on multiple quality factor scores rather than

relying heavily on one or two. Relying on multiple quality scores for classification matches with the hypothesis that the 8 scores in conjunction with one another indicate the quality of the conversation rather than just one or two quality factors. The ICE analysis paints a bleaker picture for model predictions as it shows on average most quality factors contribute towards the bad (2) or okay (1) class, and this may be caused by exhaustive substitution of quality factor values that were not commonly observed in the dataset. For instance, feature appearance values differed but rarely exceeded .90, so the model may not have accurately learned values as high or higher than .90 indicate good conversations. The SHAP charts by comparison show a more rounded impact for each quality score, where each score can push the class prediction toward good or bad classes. This is theorized to be because the modifications in each prediction during the SHAP analysis come from guaranteed real quality factor values observed in the dataset so the model should have learned how that score impacts the end classification.

The project combined two datasets, E-redial and INSPIRED, created 3 variants of a neural network using GPT2, GPT-NEO, and T5 as base models to classify recommendation explanations as good (0), okay (1), or bad (2). Conversational data was analyzed and 8 quality factors, length, readability, repetition, polarity, subjectivity, word importance, grammar, and feature appearance scores were calculated and added as inputs. The NEO variant achieved the highest prediction accuracy, likely due to its GPT2 like architecture and its more frequent use of local attention. ICE analysis showed that grammar, feature appearance, and repetition were impactful quality indicators on a local level. The SHAP analysis showed that all quality factors demonstrated the ability to impact model predictions.

CHAPTER VI.

Conclusion and Future Direction

Once a model achieves a baseline proficiency in expressing dialogue, the relevance of conventional metrics like BLEU, average entity score, and perplexity diminishes. At this juncture, the focus shifts towards metrics that align with human judgments, rather than those that assess the format or ability of the network to model language. Metrics like repetition, length, readability, word importance, grammar, polarity, feature appearance, and subjectivity are directly derived from the conversation itself and are closer to how humans think of quality markers than conventional metrics. The 8 quality factors provide a clearer understanding of the dialogue's quality and also offer easily interpretable feedback, aligning more closely with human perceptions of conversational excellence. The quality factors help to both assess conversational quality and explain the workings of models regardless of conversational paradigm and can be used as model inputs (as done here), or be used as outputs for other components of the CRS. There is a need to evolve beyond standardized metrics and incorporate more relevant data, such as those used here, to function as better fine-tuning and feedback mechanisms that mirror human quality values and can more accurately help CRS serve better recommendations to users.

Future directions for CRS and conversational AI involve leveraging retrieval augmented generation (RAG), which combines the benefits of retrieval-based methods with generative models. RAG models integrate a retriever component to fetch relevant information from a knowledge base or corpus, which is then used by the generator to produce responses. This would be similar to incorporating Zhou et al. [6] work with knowledge graphs and the work here with LLMs. Knowledge graphs introduce easily identifiable explainability pathways, where the route from user query to final recommendation(s) can be drawn for the path through the graph. Knowledge graphs can also help to combat hallucinations from LLMs. By incorporating RAG into CRS, systems can provide more contextually relevant

recommendations by leveraging the vast knowledge within LLMs and external sources such as databases or documents. This method may more accurately capture feature appearance and is likely to outperform the feature appearance scoring used in this work.

CRS engineers may also look towards utilizing cloud Services like GCP or AWS. Cloud services offer scalable infrastructure and resources for training and deploying large language models, making them accessible for CRS developers. Integrating LLMs with cloud services enables faster experimentation, deployment, and scaling of conversational systems, allowing CRS to handle larger volumes of data and serve more users efficiently.

Overall, future directions for CRS involve leveraging advanced techniques like RAG, utilizing cloud services for scalability, shifting towards human-centric evaluation metrics, integrating quality factors for better evaluation and refinement, and evolving beyond standardized metrics to serve users more effectively. These advancements aim to enhance the accuracy, relevance, and overall quality of conversational recommendations provided by CRS.

BIBLIOGRAPHY

[1] C. Gao, W. Lei, X. He, M. de Rijke, and T.-S. Chua, "Advances and Challenges in Conversational Recommender Systems: A Survey," *AI Open*, vol. 2, pp. 100–126, 2021, arXiv:2101.09459 [cs]. [Online]. Available: http://arxiv.org/abs/2101.09459

[2] X. Chen, Y. Zhang, and J.-R. Wen, "Measuring "Why" in Recommender Systems: a Comprehensive Survey on the Evaluation of Explainable Recommendation," Feb. 2022, arXiv:2202.06466 [cs]. [Online]. Available: http://arxiv.org/abs/2202.06466

[3] Z. Fu, Y. Xian, Y. Zhang, and Y. Zhang, "Tutorial on Conversational Recommendation Systems," in *Proceedings of the 14th ACM Conference on Recommender Systems*, ser. RecSys '20. New York, NY, USA: Association for Computing Machinery, Sep. 2020, pp. 751–753. [Online]. Available: https://doi.org/10.1145/3383313.3411548

[4] W. Lei, X. He, M. de Rijke, and T.-S. Chua, "Conversational Recommendation: Formulation, Methods, and Evaluation," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '20. New York, NY, USA: Association for Computing Machinery, Jul. 2020, pp. 2425–2428. [Online]. Available: https://doi.org/10.1145/3397271.3401419

[5] W. Lei, X. He, Y. Miao, Q. Wu, R. Hong, M.-Y. Kan, and T.-S. Chua, "Estimation-Action-Reflection: Towards Deep Interaction Between Conversational and Recommender Systems," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, Jan. 2020, pp. 304–312, arXiv:2002.09102 [cs]. [Online]. Available: http://arxiv.org/abs/2002.09102

[6] K. Zhou, W. X. Zhao, S. Bian, Y. Zhou, J.-R. Wen, and J. Yu, "Improving Conversational Recommender Systems via Knowledge Graph based Semantic Fusion," Jul. 2020, arXiv:2007.04032 [cs]. [Online]. Available: http://arxiv.org/abs/2007.04032

[7] K. Zhou, W. X. Zhao, H. Wang, S. Wang, F. Zhang, Z. Wang, and J.-R. Wen, "Leveraging Historical Interaction Data for Improving Conversational Recommender System," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, ser. CIKM '20. New York, NY, USA: Association for Computing Machinery, Oct. 2020, pp. 2349–2352. [Online]. Available: https://doi.org/10.1145/3340531.3412098

[8] S. Guo, S. Zhang, W. Sun, P. Ren, Z. Chen, and Z. Ren, "Towards Explainable Conversational Recommender Systems," in *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, Jul. 2023, pp. 2786–2795, arXiv:2305.18363 [cs]. [Online]. Available: http://arxiv.org/abs/2305.18363

[9] Y. Zhang, X. Chen, Q. Ai, L. Yang, and W. B. Croft, "Towards Conversational Search and Recommendation: System Ask, User Respond," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, ser. CIKM '18. New York, NY, USA: Association for Computing Machinery, Oct. 2018, pp. 177–186. [Online]. Available: https://dl.acm.org/doi/10.1145/3269206.3271776

[10] Z. Fayyaz, M. Ebrahimian, D. Nawara, A. Ibrahim, and R. Kashef, "Recommendation Systems: Algorithms, Challenges, Metrics, and Business Opportunities," *Applied Sciences*, vol. 10, no. 21, p. 7748, Jan. 2020. [Online]. Available: https://www.mdpi.com/2076-3417/10/21/7748

[11] S. E. Finch and J. D. Choi, "Towards Unified Dialogue System Evaluation: A Comprehensive Analysis of Current Evaluation Protocols," in *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, O. Pietquin, S. Muresan, V. Chen, C. Kennington, D. Vandyke, N. Dethlefs, K. Inoue, E. Ekstedt,

and S. Ultes, Eds. 1st virtual meeting: Association for Computational Linguistics, Jul. 2020, pp. 236–245. [Online]. Available: https://aclanthology.org/2020.sigdial-1.29

[12] D. Jannach, "Evaluating Conversational Recommender Systems: A Landscape of Research," *Artificial Intelligence Review*, Jul. 2022, arXiv:2208.12061 [cs]. [Online]. Available: http://arxiv.org/abs/2208.12061

[13] K. Krauth, S. Dean, A. Zhao, W. Guo, M. Curmei, B. Recht, and M. I. Jordan, "Do Offline Metrics Predict Online Performance in Recommender Systems?" Nov. 2020, arXiv:2011.07931 [cs]. [Online]. Available: http://arxiv.org/abs/2011.07931

[14] B. Wen, Y. Feng, Y. Zhang, and C. Shah, "ExpScore: Learning Metrics for Recommendation Explanation," in *Proceedings of the ACM Web Conference 2022*, ser. WWW '22. New York, NY, USA: Association for Computing Machinery, Apr. 2022, pp. 3740–3744. [Online]. Available: https://dl.acm.org/doi/10.1145/3485447.3512269

[15] A. Vultureanu-Albişi and C. Bădică, "Recommender Systems: An Explainable AI Perspective," in *2021 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, Aug. 2021, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/9548125

[16] A. Barredo Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera, "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI," *Information Fusion*, vol. 58, pp. 82–115, Jun. 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566253519308103

[17] M. Caro-Martínez, G. Jiménez-Díaz, and J. A. Recio-García, "Conceptual Modeling of Explainable Recommender Systems: An Ontological Formalization to Guide Their

Design and Development," *Journal of Artificial Intelligence Research*, vol. 71, pp. 557–589, Sep. 2021. [Online]. Available: https://dl.acm.org/doi/10.1613/jair.1.12789

[18] C.-W. Liu, R. Lowe, I. Serban, M. Noseworthy, L. Charlin, and J. Pineau, "How NOT To Evaluate Your Dialogue System: An Empirical Study of Unsupervised Evaluation Metrics for Dialogue Response Generation," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 2122–2132. [Online]. Available: https://aclanthology.org/D16-1230

[19] Z. Chen, X. Wang, X. Xie, M. Parsana, A. Soni, X. Ao, and E. Chen, "Towards explainable conversational recommendation," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, ser. IJCAI'20, pp. 2994–3000.

[20] J. Dietmar and M. Ahtsham, "End-to-End Learning for Conversational Recommendation: A Long Way to Go?" Jan. 2020, pp. 72–76. [Online]. Available: https://typeset.io/papers/end-to-end-learning-for-conversational-recommendation-a-long-45cjl4fz34

[21] S. A. Hayati, D. Kang, Q. Zhu, W. Shi, and Z. Yu, "INSPIRED: Toward Sociable Recommendation Dialog Systems," Oct. 2020, arXiv:2009.14306 [cs]. [Online]. Available: http://arxiv.org/abs/2009.14306

[22] R. Flesch, "A new readability yardstick." *Journal of Applied Psychology*, vol. 32, no. 3, pp. 221–233, Jun. 1948. [Online]. Available: https://typeset.io/papers/a-new-readability-yardstick-3f78yoqe3i

[23] J. Ramos, "Using tf-idf to determine word relevance in document queries," 1999.

[24] A. Celikyilmaz, E. Clark, and J. Gao, "Evaluation of Text Generation: A Survey," Jun. 2020. [Online]. Available: https://arxiv.org/abs/2006.14799v2

[25] D. Kelly, "Methods for Evaluating Interactive Information Retrieval Systems with Users," *Foundations and Trends in Information Retrieval*, vol. 3, no. 1—2, pp. 1–224, Jan. 2009. [Online]. Available: https://doi.org/10.1561/1500000012

[26] K. Zhou, X. Wang, Y. Zhou, C. Shang, Y. Cheng, W. X. Zhao, Y. Li, and J.-R. Wen, "CRSLab: An Open-Source Toolkit for Building Conversational Recommender System," Jan. 2021, arXiv:2101.00939 [cs]. [Online]. Available: http://arxiv.org/abs/2101.00939

[27] K. Sinha, P. Parthasarathi, J. Wang, R. Lowe, W. L. Hamilton, and J. Pineau, "Learning an Unreferenced Metric for Online Dialogue Evaluation," May 2020, arXiv:2005.00583 [cs]. [Online]. Available: http://arxiv.org/abs/2005.00583

[28] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: a method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ser. ACL '02. USA: Association for Computational Linguistics, Jul. 2002, pp. 311–318. [Online]. Available: https://dl.acm.org/doi/10.3115/1073083.1073135

[29] S. Banerjee and A. Lavie, "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*, J. Goldstein, A. Lavie, C.-Y. Lin, and C. Voss, Eds. Ann Arbor, Michigan: Association for Computational Linguistics, Jun. 2005, pp. 65–72. [Online]. Available: https://aclanthology.org/W05-0909

[30] R. Li, S. Kahou, H. Schulz, V. Michalski, L. Charlin, and C. Pal, "Towards Deep Conversational Recommendations," Mar. 2019, arXiv:1812.07617 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1812.07617

[31] C.-Y. Lin, "ROUGE: A Package for Automatic Evaluation of Summaries," in *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, Jul. 2004, pp. 74–81. [Online]. Available: https://aclanthology.org/W04-1013

[32] J. Dodge, A. Gane, X. Zhang, A. Bordes, S. Chopra, A. Miller, A. Szlam, and J. Weston, "Evaluating Prerequisite Qualities for Learning End-to-End Dialog Systems," Apr. 2016, arXiv:1511.06931 [cs]. [Online]. Available: http://arxiv.org/abs/1511.06931

[33] Z. Liu, H. Wang, Z.-Y. Niu, H. Wu, and W. Che, "DuRecDial 2.0: A Bilingual Parallel Corpus for Conversational Recommendation," Sep. 2021, arXiv:2109.08877 [cs]. [Online]. Available: http://arxiv.org/abs/2109.08877

[34] S. Moon, P. Shah, A. Kumar, and R. Subba, "OpenDialKG: Explainable Conversational Reasoning with Attention-based Walks over Knowledge Graphs," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, A. Korhonen, D. Traum, and L. Màrquez, Eds. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 845–854. [Online]. Available: https://aclanthology.org/P19-1081

[35] E. Sezerer and S. Tekir, "A Survey On Neural Word Embeddings," Oct. 2021, arXiv:2110.01804 [cs]. [Online]. Available: http://arxiv.org/abs/2110.01804

[36] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," Aug. 2023, arXiv:1706.03762 [cs]. [Online]. Available: http://arxiv.org/abs/1706.03762

[37] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," May 2019, arXiv:1810.04805 [cs]. [Online]. Available: http://arxiv.org/abs/1810.04805

[38] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, Eds. Association for Computational Linguistics, pp. 7871–7880. [Online]. Available: https://aclanthology.org/2020.acl-main.703

[39] C.-M. Wong, F. Feng, W. Zhang, C.-M. Vong, H. Chen, Y. Zhang, P. He, H. Chen, K. Zhao, and H. Chen, "Improving Conversational Recommendation System by Pretraining on Billions Scale of Knowledge Graph," Apr. 2021, arXiv:2104.14899 [cs]. [Online]. Available: http://arxiv.org/abs/2104.14899

[40] A. Goldstein, A. Kapelner, J. Bleich, and E. Pitkin, "Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation." [Online]. Available: http://arxiv.org/abs/1309.6392

[41] S. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," Nov. 2017, arXiv:1705.07874 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1705.07874

[42] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language Models are Unsupervised Multitask Learners," 2019. [Online]. Available: https://www.semanticscholar.org/paper/Language-Models-are-Unsupervised-Multitask-Learners-Radford-Wu/9405cc0d6169988371b2755e573cc28650d14dfe

[43] S. Black, S. Biderman, E. Hallahan, Q. Anthony, L. Gao, L. Golding, H. He, C. Leahy, K. McDonell, J. Phang, M. Pieler, U. S. Prashanth, S. Purohit, L. Reynolds, J. Tow, B. Wang, and S. Weinbach, "GPT-NeoX-20B: An Open-Source

Autoregressive Language Model," Apr. 2022, arXiv:2204.06745 [cs]. [Online]. Available: http://arxiv.org/abs/2204.06745

[44] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," Sep. 2023, arXiv:1910.10683 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1910.10683