Using BERT and GPT2 to classify Explanations of Movie Recommendations as Good or Bad

Joseph May
Computer Science
Middle Tennessee State University
Murfreesboro, USA
jam2ft@mtmail.mtsu.edu

***Abstract*—This study focuses on Conversational Recommender Systems (CRS) and proposes a method for classifying movie recommendations as good or bad. Traditional conversational recommendation metrics like BLEU, ROGUE, METEOR are not sophisticated enough to assess recommendation quality, and deep learning models like GPT2 and BERT give models an understanding of language and context but are not fine-tuned for conversational recommendation classification. The study utilizes the E-redial dataset and a classification model that relies on BERT embeddings and cosine similarity for recommendation quality assessment. The conversation parts of the seeker and recommender in an E-redial conversation are summarized separately, using BART. The summaries are embedded using BERT and the cosine similarity between the seeker's and recommender's embeddings is calculated with cosine similarity to classify a recommendation as good when the similarity is greater than or equal to 0.87 and classify it as bad when below the threshold. The effects of adam, rmsprop, and SGD optimizers, and BERT and GPT2 were examined to determine which combination yielded the best classification accuracy. Adam was the best performing optimizer across BERT and GPT2 models, and GPT2 outperformed BERT and achieved a maximum classification accuracy of 84%.**

***Index Terms*—BERT, BART, GPT2, Conversational Recommender Systems, CRS, Machine Learning, AI, Transformers, NLP**

I. Introduction

Conversational recommender systems (CRS) are recommendation engines that determine what to recommend to a user based on a conversation [5]. 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. 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.

CRSs are comprised of two main components, a conversation and a recommendation module. 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, a visual representation of this may be seen in Figure 1 [16]. 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, so the system can be sure of a good recommendation takes more time, but increases the likelihood that the recommendation is good. The system must also determine how much to clarify when a user's query is ambiguous. Interaction patterns for a CRS broadly fall into four categories.

The system active user passive (SAUP) model has the system ask the user questions, and the user is only supposed to 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 [4].

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 [4].

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 [4].

Finally, is the active user and a 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 [4].

The effectiveness of CRSs 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. Current metrics primarily assess if the recommendation was successful, 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 movie recommendations as good or bad.

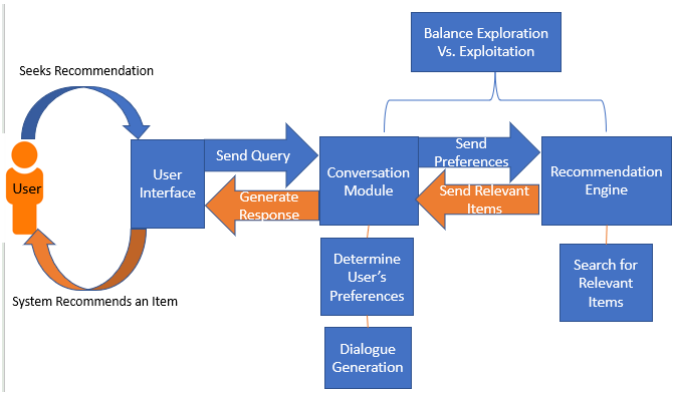

Fig. 1. A high-level diagram depicting the workflow of a Conversational Recommender System.

II. Background

As noted above there exists a gap in methods to assess the quality of a recommendation. Some common metrics are discussed below.

BLEU is a word overlap metric that looks for shared n-grams between the ground truth and a possible response [12] [11] [7]. 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.*" Because this metric is looking for shared n-grams, 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 [1] [9]. 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 [10] [11]. 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 [11]. 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.

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 Natural Language Processing (NLP) tasks, providing a dense numerical representation of words that facilitates machine learning models in understanding and processing language [13]. 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 [13]. 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 [13]. Word embeddings and the transformer architecture have revolutionized natural language processing (NLP). The Transformer architecture is a neural network architecture introduced in the paper "Attention is All You Need" by Vaswani et al [14]. 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 [14]. 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 [3] and BART [8] transformer models are used in this experiment.

Bidirectional Encoder Representations from Transformers (BERT), was introduced in "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," Devlin et al [3]. 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) [3]. Unidirectional 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 [3].

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 [8]. 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 [8]. This denoising objective encourages the model to learn robust representations by understanding the relationships between different parts of the input.

Cosine similarity is used to assign class labels in the experiment. Cosine similarity is a metric used to measure 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 close to 1 the similarity is the more similar the vectors are, 0 implies perpendiculairty 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.

The paper Measuring "Why" in Recommender Systems: a Comprehensive Survey on the Evaluation of Explainable Recommendation [2] provides a comprehensive 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. Multiple factors of explainability are critical for accurately classifying recommendations, and multiple factors are not implemented in this work, the addition of more factors is planned in the future.

Towards Explainable Conversational Recommender Systems by Guo et al [6]. addresses 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 [6]. 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 [15]. 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 [15]. 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.

The dataset used in this paper is the E-redial dataset introduced in towards explainable conversational recommender systems [6]. 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 [6]. 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.

III. Methods

A. Target Creation

The E-redial dataset does not contain classifications of good and bad recommendations. It contains 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 use to classify each conversation.

Each conversation in the E-redial dataset is broken into messages with a speaker tag, and their message to the other user. Since the data is a recording of a conversation between two people, the order of the messages is not guaranteed, nor is the number of messages sent by each user. The first speaker may be either person, and either speaker may send as many consecutive messages as they want. As an example, the recommender may send three movie recommendations as separate messages and then the seeker may respond once mentioning how they agree/disagree with each of the three recommendations, or the user may respond with three separate messages of their own.

When collating the data to generate target labels, 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 speakers contributions to the conversation. The summary IDs generated by BART are then decoded by the BART model, which returns a text 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. BART was selected for the summarization task because it was designed for abstractive summarization, and thus a more fitting choice for summary

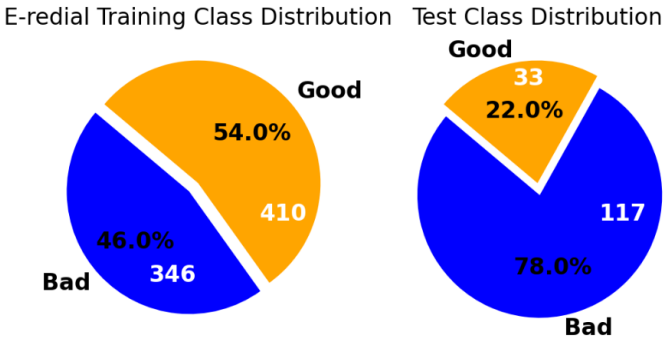

Fig. 2. A chart depicting class label distribution after target labels were generated.

than BERT which does well at many NLP tasks, but was primarily designed for word embeddings.

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. If the cosine similarity of the embeddings is greater than or equal to 0.87, that conversation is classified as a good recommendation, anything below 0.87 is classified as a bad recommendation. The BERT-based-uncase tokenizer and model were used for tokenization and the embedding process. BERT is an apt choice for word embeddings as it has bidirectional context to understand the meaning of a word in the context of the entire conversation, has a good grasp of the English language through its pretraining regime, and weights words differently based on their attention score, making it an effective tool to embed words to capture their meaning and intent. 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.

After the label generation process was finished the training set of E-redial contained 346 bad conversations and 410 good conversations, leading to a 46% bad to 54% good class division in the training data. The E-redial test set contains 150 conversations 117 of which were classified as bad, 33 were classified as good, leading to 78% of the examples in the test set being bad and 22% as good conversations, these results may be seen in figure 2. The class imbalance in the

test set is explicitly part of the dataset as the creators noted in their paper 823 of the system responses in the test set are idle with no movie recommendations, meaning the majority of conversations in the test set are just chit chat and thus not good recommendation dialogue [6].

B. Data Embedding

The data is embedded using an encoder network and relies on either GPT2 or BERT. The messages between the seeker and recommender are gathered in an array with the order of the messages preserved, and consecutive utterances are joined into a single array index. So if the recommender sends the message, 'Hello", then the message 'How may I help today?", and the seeker responds, 'Hi, I'm looking for an action movie" the conversation has the form ['hello. How may I help today?", 'Hi, I'm looking for an action movie"]. Each conversation is stored in an array of strings in this format. After each conversation has been converted to this format there are 756 training conversation string arrays and 150 test conversation string arrays. Each conversation is padded to the maximum conversation length found in the dataset so that all conversations contain the same number of string messages.

This information is then passed through the encoder network, which converts the conversation into a single string, and tokenizes the conversation. The BERT tokenizer is used for the BERT model, and GPT2 tokenizer is used for GPT2 model so that each pretrained model recieves the data in the format it was trained to expect. Once the data has been tokenized it is is passed to the pre-trained model, BERT or GPT2, to be embedded. Each conversation is converted into a tensor with input IDs and an attention mask, which get squeezed to remove unnecessary dimensions. Also in this process, the target label is converted from an integer to a tensor and is joined to the conversation embedding so that the conversation data and target information are now bundled together as (input_id,attention_mask,label).

C. Testing Procedure

The effects of optimizer choice and base pre-trained model are examined in this experiment. Three optimizers were selected for examination, SGD, ADAM, and RMSPROP. The two pretrained models selected for exaination in this experiment were BERT and GPT2. Due to memory constraints on the GPU batch sizes had to be kept relatively small batch sizes of 4 were selected for training, and batch sizes of 5 were used for testing. The experiment was run independently 10 times, and went for 60 epochs.

The embedded data is passed into a classifier network which takes the conversation embeddings and target labels. The classifier network uses cross-entropy loss to minimize the distance between model predictions and the actual classification of the conversation. In the forward pass the model passes the input IDS and attention mask embeddings through BERT/GPT2 to obtain and extract BERT's pooled output or the final layer of GPT2, which are fixed-size embeddings that represent the entire conversation. The pooled output is fed through a fully connected layer set to the hidden layer size of BERT or GPT2 for each respective model. The resulting logits of the dense layer represent the raw scores for each class, enabling the model to make predictions and calculate the loss. The accuracy and loss of the training and testing datasets are recorded for further analysis.

IV. Results

Each model was trained for 60 epochs and run independently 10 times. The results of the BERT model may be seen in Figure 3. In Figure 3 the adam optimizer performs the best, SGD performs 2nd best and rmsprop performs the worst by far. This matches expectations as adam has become a default choice in deep learning for its robustness and combination of the best elements of SGD and rmsprop. The large standard error bands are likely due to a combination of few quality indicators, class imbalance in the validation set, and negative transfer learning. The only quality indicator used to classify recommendations in the experiment was cosine similarity, other factors such as polarity, length, and others may be required to increase the consistency of the model. The creators of E-redial left the validation set with conversations that were mainly chit-chat and not recommendation-based, which ultimately meant that the class distribution for the experiment was heavily skewed towards the negative even though in training the model would ideally come to expect a roughly equal distribution. Altering the Validation set to not be as heavily skewed may also reduce the standard error. The final reason for high standard error is theorized to be negative transfer learning from BERT. BERT was trained on a far larger corpus than E-redial and the adjustment of the pre-trained weights appears to make the model erratic rather than improve accuracy.

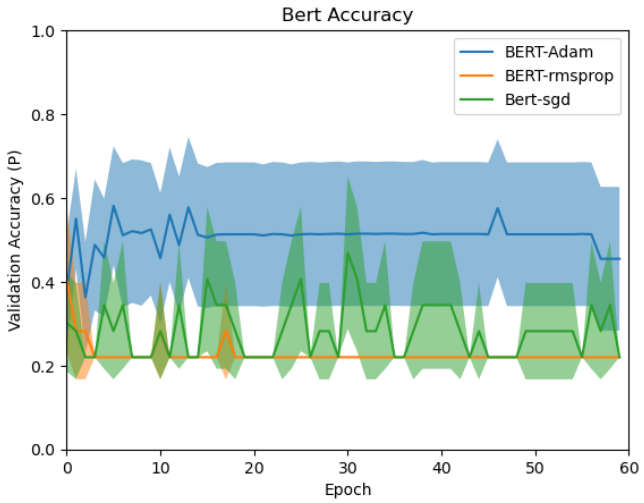

Fig. 3. A chart depicting the accuracy ratings of the BERT model with different optimizer choices

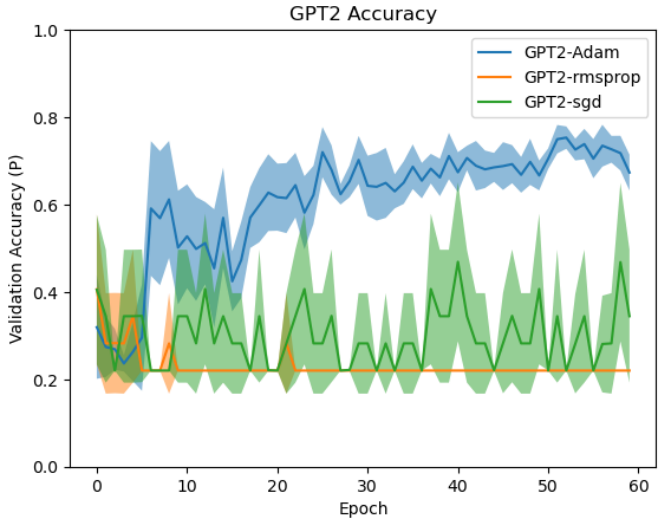

Fig. 4. A chart depicting the accuracy ratings of the GPT2 model with different optimizer choices

The results for the GPT2 model may be seen in Figure 4. The trend of results is similar to BERT. Adam is the best optimizer, then SGD then rmsprop, and it is believed to be caused for the same reasons. More notably for GPT2 the standard error is much smaller the adam optimizer. This is believed to be a side effect of transfer learning from GPT2. Due to hardware limitations, the weights of the GPT2 model had to be frozen, whereas the BERT model weights were adaptable through backpropagation. It is believed that in addition to the architectural differences of GPT2 that the freezing of the weights contributed to GPT2's clear performance wins, and that further testing with frozen BERT weights may close the accuracy gap between the two models. Similar to BERT it is likely that incorporating additional quality factors outside of cosine similarity, or a re-balancing of the validation set could reduce the standard error and improve accuracy.

V. Discussion

This paper used two powerful pre-trained models, BERT and GPT2, in order to classify conversational recommendations as good or bad. Unlike works mentioned previously, which emphasize the importance of using multiple factors to assess conversational recommendation quality, this work used only relevance scores via cosine similarity of the seeker and recommender's conversational portions to label conversational recommendations as good or bad. As expected the usage of BERT was effective in the downstream classification task, and achieved a fairly high accuracy rating for conversational recommendations, however, GPT2 outperformed BERT on average. This is most likely due to negative transfer learning in the BERT model. Due to memory constraints on the GPU the GPT2 model weights had to be frozen, allowing for more successful transfer learning with accuracy gains coming from adjustments to the final linear layers of the model.

Following the examples of the papers above, incorporating further factors of recommendation quality in particular polarity, word importance and length are likely to improve classification accuracy further. Adding additional complexity to the model by adding more dense layers and transformer blocks is likely to also improve accuracy. Nevertheless, it is encouraging that a pre-trained model incorporating only a single quality factor for recommendations was able to achieve successful classification results.

Future work should focus on increasing the complexity of the classification model by adding additional layers, adding in more robust speaker encodings to denote the structure of the conversation, including more quality factors for conversational recommendation, and adapting to incorporate different modalities of data such as knowledge graphs, or picture data associated with recommended items, adapting the problem from binary to multiclass to capture a more diverse range of recommendation quality, and by incorporating more diversified conversational data such as that from the INSPIRED dataset, or recommendation conversations not centered around movies.

VI. Acknowledgements

Thanks to the reviewers for their comments which have improved this paper

References

[1] Banerjee, S. and Lavie, A. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In Goldstein, J., Lavie, A., Lin, C.-Y., and Voss, C., editors, *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72. Association for Computational Linguistics.

[2] Chen, X., Zhang, Y., and Wen, J.-R. Measuring "why" in recommender systems: a comprehensive survey on the evaluation of explainable recommendation.

[3] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding.

[4] Fu, Z., Xian, Y., Zhang, Y., and Zhang, Y. Tutorial on Conversational Recommendation Systems. In *Proceedings of the 14th ACM Conference on Recommender Systems*, RecSys '20, pages 751–753, New York, NY, USA, September 2020. Association for Computing Machinery.

[5] Gao, C., Lei, W., He, X., de Rijke, M., and Chua, T.-S. Advances and challenges in conversational recommender systems: A survey. *AI Open*, 2:100–126, January 2021.

[6] Guo, S., Zhang, S., Sun, W., Ren, P., Chen, Z., and Ren, Z. Towards explainable conversational recommender systems. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2786–2795.

[7] Hayati, S. A., Kang, D., Zhu, Q., Shi, W., and Yu, Z. INSPIRED: Toward Sociable Recommendation Dialog Systems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8142–8152, Online, November 2020. Association for Computational Linguistics.

[8] Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J., editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880. Association for Computational Linguistics.

[9] Li, R., Kahou, S., Schulz, H., Michalski, V., Charlin, L., and Pal, C. Towards Deep Conversational Recommendations, March 2019. arXiv:1812.07617 [cs, stat].

[10] Lin, C.-Y. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81. Association for Computational Linguistics.

[11] Liu, C.-W., Lowe, R., Serban, I., Noseworthy, M., Charlin, L., and Pineau, J. 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*, pages 2122–2132, Austin, Texas, November 2016. Association for Computational Linguistics.

[12] Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318. Association for Computational Linguistics.

[13] Sezerer, E. and Tekir, S. A survey on neural word embeddings.

[14] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need.

[15] Wen, B., Feng, Y., Zhang, Y., and Shah, C. ExpScore: Learning metrics for recommendation explanation. In *Proceedings of the ACM Web Conference 2022*, WWW '22, pages 3740–3744. Association for Computing Machinery.

[16] Zhang, Y., Chen, X., Ai, Q., Yang, L., and Croft, W. B. Towards Conversational Search and Recommendation: System Ask, User Respond. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 177–186, Torino Italy, October 2018. ACM.