Open Question Identification Using Statistical Models

Cameron Plume
CS 4120
Written For Professor Amir Tahmesabi, PhD.

Abstract

In many kinds of conversational formats, from meetings to chat rooms, people ask questions that go unanswered. Developing a natural language model to identify unanswered questions in conversational natural text could have use cases spanning meeting transcription technologies, customer support, and email application extensions. In this study, an attempt to make a two stage pipeline for open question identification was made. First, a model to identify phrases as questions or non questions was developed. Next, a model was created with the aim of identifying whether a given phrase answers a given question. Finally, these models were combined, and using the patterns of answer occurrences, attempted to classify unanswered questions in conversational text. While individual models were performant in their tasks with cleaned data, the full model pipeline was largely unsuccessful at handling open question identification due to compounding error, and the presence of typos, slang, and otherwise convoluted text.

Introduction

The idea for this project was formulated by a former consultant. In the management consulting field, junior consultants and interns are often tasked with taking verbatim notes, then generating summaries and follow up items. "Follow up" items are being defined in this context as issues that are raised during a meeting that need additional action or research. They often appear in the form of a question that has not been answered.
The goal of this project is to create a statistical model capable of identifying unanswered questions in a conversational free text corpus. Such a model would have many applications, some of which are described here:

Customer Support Quality Assurance: Such a model could be helpful in evaluating the performance of both human and automated customer support services, allowing firms to identify when user questions are not properly addressed

Online Forums: Unanswered question flagging could be useful in online chat forums, flagging open threads that have not been answered. This can be useful especially in educational forums, and for content moderation.

Meeting Transcript Follow Ups: This model could be implemented on top of existing speech-to-text technologies, allowing users to identify conversational items that were not answered or properly addressed.

Email Communication: This model could be used as a personal tool to ensure responses to emails and other forms of personal communication address all of the information requested.

Reinforcement Learning for Question Answering Models: This technology could be used to evaluate the performance of Question Answering systems in modern LLM's.

It is necessary to clarify a few key points of the use case for this model.

First, due to the original use case as a supplement for speech-to-text technology, punctuation is not used. Question marks are a highly significant feature for question recognition, but this study focused on informally written and transcribed corpuses, where punctuation is unlikely to be present or helpful.

Next, due to the nature of proposed use cases, Type 2 or "false negatives" should be minimized. It is best to flag potential items for follow up that had previously been resolved, rather than to miss potentially unresolved questions. This affected the strategy used in the final model development.

Due to the potential use cases of the model, it is most important to be able to "flag", or predict, that an unanswered question has in fact gone unanswered. Therefore recall will be one of the central evaluative metrics for this model. Ideally, when implemented in conjunction with a user, this model will be capable of flagging phrases for review, and should attempt to avoid false negatives.

Finally, this model makes no attempt to validate the correctness or value of an answer, rather just that an answer was given at all.

Related Works

There is a fair amount of previous work in this area, as well as many closely related problems.

A variation of this problem has been extensively addressed in the training of question answering systems. Often, these models seek to quantify the relevance of an answer to a question as a loss metric for QA models. Usually continuous values are used, rather than binary classification of relevant or not relevant. The highest performing QA systems are built off of transformer based LLM's like BERT and GPT.

In addition to answer relevance, a similar problem has been studied identifying similar and identical questions. This is an important issue for blog sites like Quora, which seek to combine similar and identical question posts in order to allow more thoughtful discussion under a single thread. The best performing models for these tasks are once again deep learning based, and largely use fine tuned transformer models.

Question identification is a less studied problem, largely because it is mostly trivial in formal and punctuated text. However some work has been done in this field as meeting transcription technologies seek to add more accurate punctuation, among with some other use cases.

Existing Data Sets

A key issue for this project was acquiring reasonable datasets for the scale of this task. In the end, the following sets were used, augmented by manual labeling and data generation using existing LLM models.

Question Data Set [1]: The SimpleQuestion dataset provided by hugging face was used. It consists of 108,442 natural language questions, paired to tuples in the format (subject, relation, object). Due to the lack of natural language answers, only the answer was used. A subset of 4000 questions was selected for runtime considerations.

Discord Thread Dataset [2]: The DISCO dataset of discord conversations was used. Specifically, 700 messages were extracted from a channel with a theme of "Python". The dataset also included thread information as well as user data, which was disregarded. Messages were stored in the dataset sequentially, which allowed timestamps also to be disregarded

Data Generation

After the collection of the above datasets, augmenting data was still needed.

Question-Answer Pairs: In order to properly build this model, relevant answers to questions were needed. These were generated using the OpenAI API. Using the prompt: "I am going to give you a question. Please reply with a single sentence that could be an answer to that question", 4000 answers were generated as pairs to the SimpleQuestion dataset. In order to create a negative class, half of the answers were "offset" by 1, in order to create false matches.

Discord Manually Labeled Set: In order to build the full demo model, it was necessary to have a conversation with labeled question-answer pairs. This was done manually for 700 entries, and some errors are likely. Three columns were added to the dataset by hand:

- question: {1: the message contained a question, 0: the message did not contain a question}
- answered: {1: the message was a question and had been answered, 0: the message was a question and had not been answered, NULL: the message was not a question}
- answer_index: {number: the index of the message that answered this question, NULL: the message is not a question}

The messages in this set are sequential, in the order that they were sent.

Question and Non Question Sets: In the building of the question recognition model, generated answers were used as the negative class.

Methods

Question Recognition Model

The first necessary piece of an open question identification model is to identify if a piece of text is a question. I used the previously described Question Answer Pairs dataset. Answers were extracted then concatenated, giving 8000 feature sentences with 50% as questions. It had the following structure:

Text (String, Feature)	is_question (1: question, 0: not question)

Importantly, the use of answers as the negative class allowed statements regarding similar topics to be present in both question and non question cases. This design decision reduces overfitting of terms only found in questions that had no effect on the grammatical nature of the sentence, and additionally was designed to help the model to perform well on corpuses that repeatedly discussed related topics.

First, data was split into a training and testing set using the sklearn library. An 80% train, 20% test set was used, resulting in 6400 and 1600 rows respectively.

According to previous works, the most statistically significant factor in the recognition of a question from free text is the presence of question words, including "who, what, where, when, do, is".

Therefore, in the creation of a model, TF-IDF was a clear choice as a method of encoding sentences. Given the nature of question words in the identification of questions, stop words were not removed. Sentences were tokenized through first the removal of punctuation (as the presence of question marks), then word tokenization and stemming using a Porter Stem.

For TF-IDF vectorization, TfIdfVectorizer provided by the sklearn library was used.

TF-IDF vectors were fitted on the training set only, and the test set was vectorized using a transform call to the sklearn vectorizer.

After TF-IDF vectorization, a Logistic Regression model was trained to generate predictions. TF-IDF vectors were used as features, and the target variable was 'question'. The logistic regression classifier used was LogisticRegression provided by the sklearn library.

Question Recognition Results and Analysis

Results:
The following confusion matrix was generated in analysis of the performance of this model

Class	Precision	Recall	F1 Score	Support

0 (not question)	0.98	0.98	0.98	798
1 (question)	0.98	0.98	0.98	804
Accuracy	-	-	0.98	1600
macro avg	0.98	0.98	0.98	1600
weighted avg	0.98	0.98	0.98	1600

Of 1600 predictions, 1566 phrases were correctly predicted, giving an overall accuracy of 0.97875.

The following is a list of the 17 statements that were incorrectly identified as questions (Type 1):

1: gridiron is a type of sports book
2: lebanon is in the eastern european time zone eet which is utc2
3: the name of the song is the collection
4: one famous person who died from pneumonia was former president george washington
5: westlifes studio album is named westlife
6: now thats what i call music 18 was released in 2005
7: the developer who designed asterix and the great rescue is sega
8: now thats what i call music
9: people who are exposed to extremely high temperatures or prolonged heat exposure can die from hyperthermia
10: meitner is named after the austrian physicist lise meitner
11: one player who is in the rugby league is sonny bill williams
12: peters was born in koldenbüttel which is located in schleswigholstein germany
13: bob dylan is a musician who plays folk music
14: the costume designer for the film a modern hero was insert name here
15: the featured song from evolution is girl named drooler
16: a person who follows veganism avoids consuming animal products
17: the editor of the die softly reissue was insert name

The following is a list of the 17 questions that were incorrectly identified as non-questions (Type 2):
1: is punk rock rarities a album or dvd
2: how is the drug nefazodone hydrochloride administered
3: is nightsongs a drama or comedy
4: name a 2002 indian tamil romantic biographical film film written and directed by vasanthabalan
5: how is venoforce i dosed
6: is feel the noise of liar liar considered a drama
7: steve hackett wrote this song
8: how is a bupivacaine hydrochloride 5 injectable solution classified
9: is one to one pop music or rock
10: name a year the boston celtics won the nba finals
11: the udigisels religion can be said to be

12: how was joe dirt car released
13: is the the unusual suspects a fantasy or mystery
14: when did rugby union stop being an olympic sport
15: name of album by the alternative metal band drowning pool
16: name an asteroid discovered on may 11 1985
17: is carlyle chalmers male or female

Analysis:
Overall, this model performed exceptionally well. Near 98% accuracy indicates a very strong model, capable of making accurate predictions on the test data.

In addition to high accuracy, recall for identifying questions was 0.98, indicating the logistic regression model was excellent at correctly predicting questions as questions, reducing the probability of incorrect predictions in downstream models.

Of the 17 Type 1 errors, all 17 contained question words of some kind. Therefore, it seems that the regression model was highly sensitive to question words and susceptible to Type 1 errors for non-questions that contain them.

The cause of Type 2 errors was more varied. 3 were imperative questions, and started with verbs. The addition of a semantic model could help to address these errors, as currently part of speech tagging is not considered in classification. The rest of the Type 2 errors did in fact contain question words, and were likely misclassified due to overfitting to very specific terms shared by the corresponding answer in the corpus. This could likely be mitigated by using a larger training corpus.

Question Answer Pairing Models

As the second task in an open question flagging model, it is necessary to determine whether a phrase is a possible answer to a given question. A corpus was created for this set using the previously generated Question Answer Pairs set. Half of the answers were offset by a row, in order to create false pairings. The resulting dataset had the following structure:

Question (String)	Answer (String)	Is_pair (1: pair, 0: not pair)

The full dataset had 4000 rows, 50% of which were correct pairings.

The data was then divided into 80% train, 20% test splits, sized 3200 and 800 respectively.

Based on previous works, TF-IDF was again used for text representation. Both questions and answers from the training set were used in a fit and transformation of TF-IDF vectors, using the TF-IDF vectorizer provided by the sklearn library. Stop words were removed using the english stop words provided by nltk. Words were stemmed using a Porter Stem algorithm before vectorization.

After vectorization, the cosine similarity of question and answer vectors was calculated, and added to the original DataFrame in a similarity column.

Next, a logistic regression model, provided by the sklearn library, was fitted using the training dataset.

Both discriminative and generative predictions were made, using predict and predict_proba methods provided by sklearn's logistic regression implementation.

Question Answer Pairing Results and Analysis

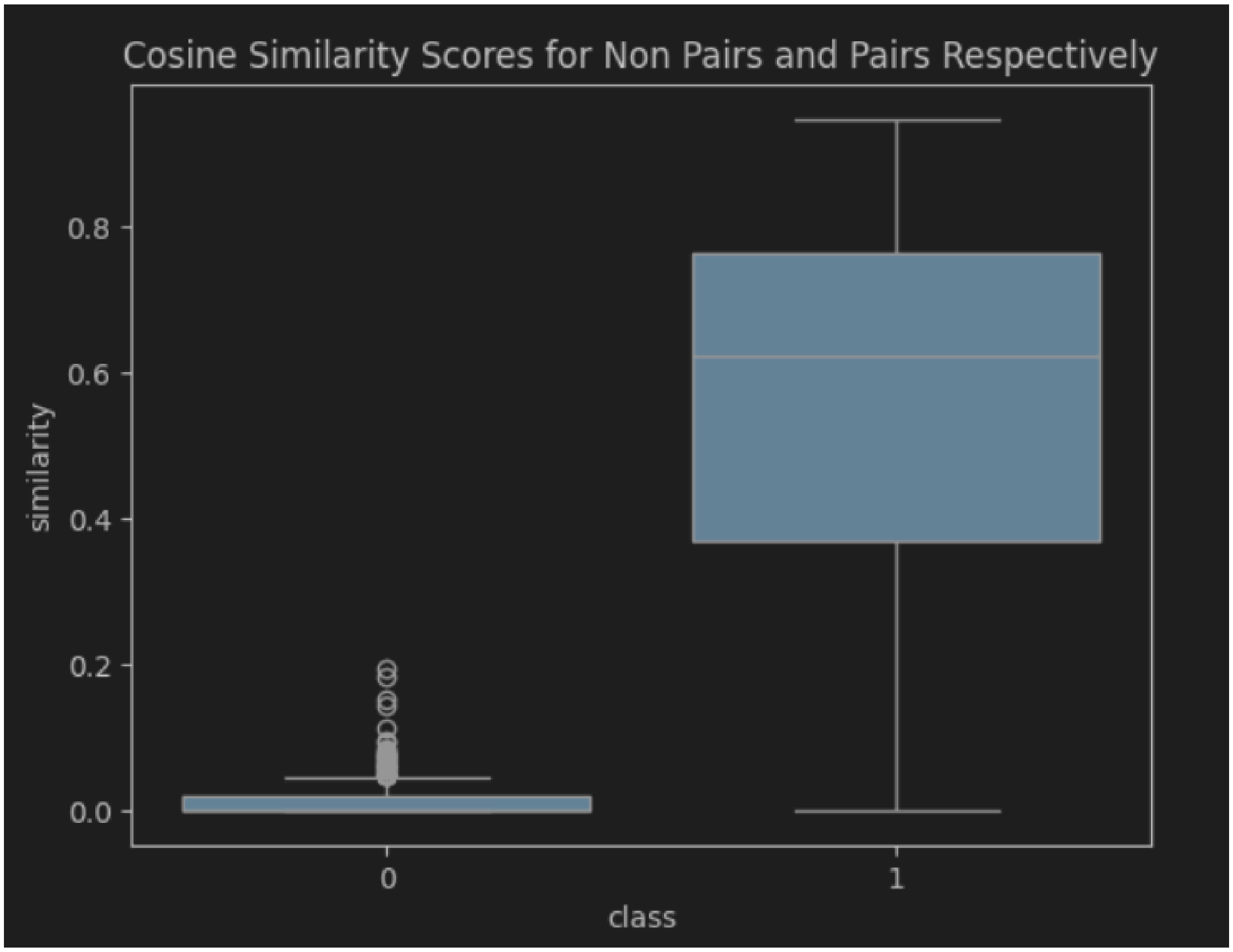
Discriminative Results:
Analysis of the discriminative predictions created by the logistic regression model yielded the following classification report over 800 predictions:

Class	Precision	Recall	F1 Score	Support
0 (Non-Pair)	0.84	0.99	0.91	406
1 (Pair)	0.99	0.80	0.89	394
accuracy	-	-	0.90	800
macro avg	0.91	0.90	0.90	800
weighted avg	0.91	0.90	0.90	800

Generative Results:
Brier score was used as the method of evaluating probabilistic results. The logistics regression model gave a Brier score of 0.086 over the 800 predictions.

Analysis:
Both models performed generally very well on the Question and Answer pairing task. Especially important to the intended use case was the recall for non-pairs in the discriminative model. 0.99 implies very very few incorrectly predicted non-pairs. 0.086 is also a very low Brier score, implying a high degree of accuracy in predicted probabilities.

The reasoning behind such high accuracy can be seen through the following box plot of cosine similarities for each class:



As shown in the graph, cosine similarities between true pairs were generally much larger. This distribution also explains the error skew towards Type 2, demonstrated by low recall for true pairs. Some pairs shared very few, or even 0 tokens, and were therefore classified as non-pairs. These errors could likely be mitigated by incorporating a thesaurus into TF-IDF vectors, as proposed in some information retrieval models. In addition, some knowledge of semantics may improve this model by more accurately predicting short answers, in cases like yes and no questions.

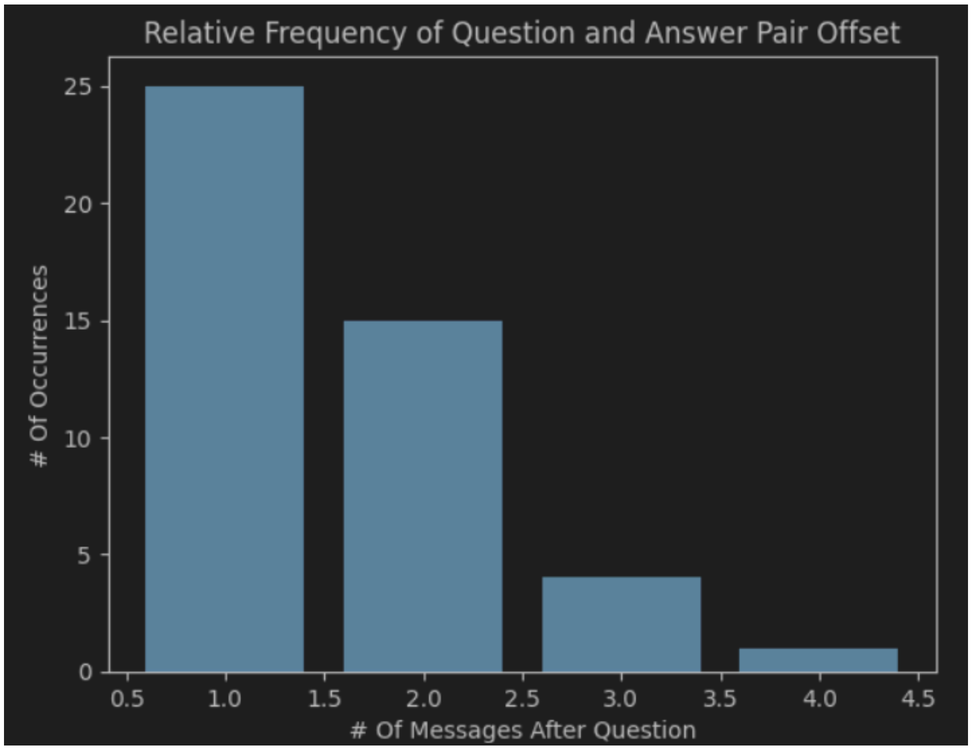
Despite these shortcomings, this is a very successful model for the described use case. It does an excellent job capturing nearly all non-pairs correctly.

Open Question Flagging Model and Demo

The Discord Manually Labeled set was used for this model. Column descriptions can be found in the dataset description. As random sampling was not possible, a train test split was created manually, using messages from distinct time periods. 500 messages were considered in the training set, and 200 in the test set.

This model requires two key tasks.

First, all questions contained in the message corpus must be identified.
Next, it must be determined if a question has been answered. This task requires some additional consideration, as not every message in the set should be considered as a possible answer. Any message sent before an identified question should not be considered as an answer. In addition, in most cases, questions are answered soon after they are posted. Therefore, using the answer_index column, the index difference between each answered question and its answer was calculated. The following distribution was found for this corpus:



As seen in this graph, all answers were found within the next 4 messages from the original posting of a question, with an inverse frequency over time.

For question identification, the logistic regression model for question recognition was used. All predicted questions were tagged.

Next, it was necessary to identify if a question had been answered, given that it was a question. The following two strategies were used.

Strategy 1: OR
Observationally, it can be seen that no questions in the training set were identified beyond 4 proceeding messages. Therefore, only the 4 messages were considered in this strategy, with equal weightings. Cosine similarity scores were generated using the vectorizer generated in Question and Answer Pairing. Each question was then compared to its 4 following neighbors, and 4 predictions were made using the discriminative model from Question and Answer Pairing. If any of these 4 predictions were positive, the

question is predicted as “answered” [value: 1]. In other words, the logical “or” was applied to the 4 predictions. If none of the predictions were positive, the question is predicted as “unanswered”[value: 0].

Strategy 2: Expected Value

In the second strategy for classification, the frequency of distance is used as a probability density function, and expected value is used to make the final prediction. Probability predictions for each of the 4 following messages were made, then multiplied by the proportion of answers found in that position in the test set. The sum of these four values was then calculated. If the sum exceeded 0.5, the question was predicted as “answered”.

In both cases the final prediction of whether a question qualified as an unanswered question was computed by calculating if a question was both predicted as a question, and no answer was found

Open Question Flagging Results

Question Identification:
On the test set, question recognition yielded the following results:

Class	Precision	Recall	F1 Score	Support
0 (non-question)	0.90	0.92	0.91	174
1 (question)	0.36	0.31	0.33	26
accuracy	-	-	0.84	200
Macro avg	0.63	0.61	0.62	200
Weighted avg	0.83	0.84	0.83	200

Given the prediction of a question, has it been answered:
Given that a phrase was predicted as a question, after the execution of the generative method of prediction the following results were recorded:

Class	Precision	Recall	F1 Score	Support
0 (Unanswered)	0.82	0.82	0.82	17
1 (Answered)	0.40	0.40	0.40	5
accuracy	-	-	0.73	22
Macro avg	0.61	0.61	0.61	22
Weighted avg	0.73	0.73	0.73	22

Given that a phrase is a question, has it been answered:
NOTE: This statistic was calculated using the training set, and it was less relevant to the final model.

Class	Precision	Recall	F1 Score	Support
0 (Unanswered)	0.60	0.72	0.65	43
1 (Answered)	0.67	0.53	0.59	45
accuracy	-	-	0.62	88
Macro avg	0.63	0.63	0.62	88
Weighted avg	0.63	0.62	0.62	500

Goal prediction: Identifying Unanswered Questions in the Test Set

Class	Precision	Recall	F1 Score	Support
0 (Not an Unanswered Question)	0.96	0.93	0.94	189
1 (Unanswered Question)	0.18	0.27	0.21	11
accuracy	-	-	0.89	200
Macro avg	0.57	0.60	0.58	200
Weighted avg	0.91	0.89	0.90	200

Open Question Flagging Analysis

Overall, each stage of this model performed at a lower level than expected.

In question identification, while accuracy stayed at 84%, recall of true questions fell to just 31%. The capability of the Question Recognition Model to identify questions in the conversational corpus fell dramatically. This was likely for a combination of reasons, all related to data cleanliness. In contrast to the initial training sets, the conversational corpus was full of slang, typos, and simple phrases followed by question marks. The combination of these factors, and lessened use of question words in the conversational corpus, significantly affected this model's ability to make positive predictions.

Question and Answer pairing also was far less effective. When evaluating phrases that were known to be questions, this model was 83% accurate. However, it had just 0.40 recall on positive identifications. This is likely low enough to make it unusable for any meaningful use case. Despite this, it was excellent with recall of unanswered questions, which did fit with the initial goal of the project.

When evaluating phrases that were predicted to be questions, the model performed similarly across both classes, with an accuracy of 62%. Importantly, these statistics were drawn from the set used in answer index pdf calculation, instead of the final test set.

These lower performances were likely due to similar confounding factors as the Question Recognition model. In addition, when attempting to predict answered questions based on predicted, rather than given question phrases, there is a compounding error effect.

Finally, the central prediction of identifying unanswered questions in a conversational corpus yielded similarly disappointing results. Of the 17 actual unanswered questions in the corpus, 3 were correctly identified. 11 were flagged as unanswered questions. This resulted in a recall of just 0.27, which was the central metric this model was hoping to optimize.

The correctly flagged unanswered questions from the test corpus were:
1: What was meant by that
2: What library
3: or I should throw an exception telling the that user didn't enter correct input

Proposed Future Work

While the tested format for this model was focused on short form messaging common in online platforms, these methods could prove to be more effective in other formats like email and other kinds of conversational formats where typos, slang, and other confounding factors are less common.
The model described in this paper focused on term frequency for question identification. However, semantics and part of speech identification can be helpful in many ways for question and answer pairing. Yes/No, quantitative, and qualitative questions can be identified using these strategies, which can inform the sorts of phrases that can be considered as possible answers.
Thesauruses and other sorts of topic identification are also tools that could yield better results in this area. This would allow synonyms to be considered. As well as answers with no term overlap. For example, this could help pair the question “What year is it” with “2023”.
Finally, deep learning techniques can be applied to these sorts of problems. As mentioned in the related works section, neural networks have been widely applied to problems like Quora’s question duplication, and have been shown to be effective.

Conclusion

Through the use of term based statistical methods, effective models for cleaned datasets were able to be created for both question identification and question-answer pairing. These models proved not to be extensible to the task of identifying open questions in conversational text, due to the use of slang, typos, punctuation, and brevity common in meeting transcriptions and online conversation

References

Hugging Face. "SimpleQuestions v2." Hugging Face Datasets. https://huggingface.co/datasets/simple_questions_v2.

K. M. Subash, L. P. Kumar, S. L. Vadlamani, P. Chatterjee and O. Baysal, "DISCO: A Dataset of Discord Chat Conversations for Software Engineering Research," 2022 IEEE/ACM 19th International Conference on Mining Software Repositories (MSR), Pittsburgh, PA, USA, 2022, pp. 227-231, doi: 10.1145/3524842.3528018.

Dami, Maximilian. "Quora Question Pairs Competition Results." Kaggle, 2017, https://www.kaggle.com/competitions/quora-question-pairs/discussion/34355.

Othman, Nouha, Rim Faiz, and Kamel Smaïli. "Enhancing question retrieval in community question answering using word embeddings." Procedia Computer Science 159 (2019): 485-494.

Pedregosa, F., et al. "Scikit-learn: Machine Learning in Python." Journal of Machine Learning Research, vol. 12, 2011, pp. 2825–2830.

Appendix

Submission Notes:
A virtual environment file has been submitted, which should allow Jupyter Notebooks to be run without additional setup.

Resource Locations:
The demo notebook and final model results is located at
demo_pipeline_on_test_conversation/demo.ipynb

The development of the question identification model is located at
q_or_not_q_classification/tf_idf_attempt.ipynb

The development of the question answer pairing is in
question_answer_pair_matching/create_q_a_pair_models.ipynb

EDA and initial statistics for the demo pipeline is located at
question_flagging/question_flagging_model.ipynb

Models and vectorizers are stored in the models directory in pickle format

Data and data acquisition scripts are stored in the data_inspection directory

Unit tests are stored in the test directory

Unit Testing and Function Declarations: I initially designed my project using a series of .py files, and made calls to these files in my notebooks. Many of these functions were shared across different notebooks. Unfortunately, this caused issues when I attempted to load in my saved models and vectorizers in my ensemble model and demo using joblib and pickle ORM libraries. I was unable to find an intelligent fix, and ended up copying the function definitions to the top of several notebooks, and again in my unit testing file. I recognize this is not ideal from a style perspective, but the functions are fully tested.

Final Output: I spent a lot of time on this project attempting to focus on abstractive text summarization, as this was my original project goal. I chose not to focus on these attempts during this report because I ultimately failed to create a working model of really any kind. I want to emphasize that this was not due to a lack of effort or time put into the problem. I faced problems finding reasonable datasets, I was limited by compute, and I faced numerous bugs in my implementation due to poor python programming skills and struggles understanding some of the libraries needed to effectively build these models. Lack of skills, yes, lack of effort, no. I am happy to provide further context on these efforts if they would be helpful.

Despite my ultimate failure to build the project I wanted, I found this exercise to be remarkably productive, and I learned a lot. Thank you for all the time and effort you put into this course, it was deeply engaging and insightful.