# Direct Supply PTAC Diagnostic Natural Language Model

Parker Splitt, Jarred Kohout, Collin Quinn, Keegan Rhodes, Andrew Ruswick

Milwaukee School of Engineering and Direct Supply

CS 3310: Data Science Practicum

Dr. John Bukowy

15 May 2022

## Abstract

Direct Supply distributes many products including Packaged Terminal Air Conditioner (PTAC) systems to senior living facilities, and they are committed to supporting senior care everywhere. To improve customer experience and reduce burden on customer support, we have implemented an AI chatbot to answer PTAC diagnostic questions for a quick and easy solution to common maintenance problems. This study investigates several stages common to AI chatbots, including scraping websites and PDF manuals for truth data, exploring strategies to validate textual outputs, and experimenting with different models to produce an intelligent AI chatbot. Our solution uses the recently developed GPT-3 natural language model to implement a virtual agent that answers PTAC maintenance questions for engineers and technicians. From our study, we concluded that a model using data sourced from scraping owner's and service manual PDF documents produces output that better aligns with the expected output in our validation strategy than using data sourced from scraping publicly available websites or a combination of both. In addition, our experiments found that BERT vectorization gives a wider range of representation for sentences than OpenAI's embeddings. Finally, we make the recommendation to use the Ada search engine and the Curie model output engine for a model that balances cost and quality of outputs.

## Client Description

Direct Supply was founded in Milwaukee, Wisconsin in 1985 and has grown to be an employee-owned company that specializes in providing equipment, eCommerce, and services to healthcare organizations and the senior living industry. The focus of this project is on Direct Supply's business relating to Packaged Terminal Air Conditioner (PTAC) distribution and maintenance in the senior living industry.

Direct Supply specializes in four major lines of business: Equipment & Furnishings, Digital Small Storage Interconnect (DSSI), Aptura, and The Equipment Lifecycle System (TELS). Direct Supply supplies equipment and furniture to healthcare organizations and senior living facilities. Direct Supply Equipment & Furnishings is the company's flagship division, supplying senior living facilities with healthcare equipment, clinical equipment, rehabilitation equipment, foodservice equipment, maintenance equipment, and furniture. DSSI is an eProcurement system that links more than 10,000 senior living facilities to their supply chain and moves about $3 billion worth of product per year. Aptura offers services including interior design, foodservice design and furniture, fixtures, and equipment (FF&E) procurement, focusing solely on the senior living industry. Finally, Direct supply hosts an online building management platform called TELS. This platform, accessed from https://www.tels.net/, serves as an online portal that helps to manage life safety, asset management, and maintenance challenges. TELS allows for submitting work orders for maintenance requests, where an appropriate technician will be deployed to solve problems involving products serviced by Direct Supply.

Direct Supply strives toward aiding seniors towards a better quality of living while they are staying in but not limited to assisted living homes. One way in which they wish to achieve this relates to the maintenance of their PTAC systems in nursing homes. Currently, nursing home technicians have no way of repairing a Direct Supply PTAC system without first consulting Direct Supply's own repair technicians. Direct Supply wants to simplify this process by supplying assisted living homes with some sort of "middle-man" in the form of a natural language model that can assist nursing home technicians by providing answers to questions asked by the users regarding maintenance of their PTAC systems.

Ultimately, this will lead to Direct Supply spending less money on sending technicians out to fix easier problems and provide a much smoother experience on the end of the nursing homes. Thus, reducing the cost of operation and increasing customer satisfaction / user experience.

Direct Supply's general line of contact is (800) 634-7328.

The sponsor of this project is Gokula Mishra, and his email is Gokula.Mishra@directsupply.com.

## Hypotheses

### Hypothesis I

Since we are forming our own dataset, we will be looking at numerous data sources. These sources include publicly available websites as well as owner's and service manuals in the form of PDFs. In order to better focus our data collection strategies, we must be able to decide which source gives us more consistent data. After initially analyzing PDFs and websites, we have hypothesized that websites will have a higher variance in quality. This is because the manual PDFs come straight from the manufacturers of the PTAC systems and likely will not have the same amount of bias or misinformation that could be present in a webpage. Therefore, our first hypothesis is the following:

1. Data scraped from the web will have a higher variance of quality compared with data scraped from the owner's and service manuals.

### Hypothesis II

While going through the data obtained from our initial web scraping methods, we noticed that there is a large amount of irrelevant information included in certain HTML elements. More specifically, there seems to be anecdotally more irrelevant information in HTML elements such as lists compared to paragraph and div elements. Upon making this observation, we wanted to observe if there was a way to concretely explore the relationship, if any, between the quality of data obtained from certain HTML elements compared to others. Thus, our second hypothesis is the following:

2. Information obtained from the paragraph and div elements of HTML structure will yield data of higher quality compared to data obtained from the other HTML elements.

### Hypothesis III

When working on this project, we thought it would be important to consider how the data gathered may affect our resulting model along with the end user's experience. Further, it is important to consider if the end user would like to have their interaction with the conversational agent to be more formal or informal; this is what led to the third hypothesis. We are under the assumption that information obtained from PDF sources will yield data that is more formal in tone compared to data mined from a website. This prompted us to consider methods of definitively testing to see if there is a quantifiable difference between the two. Our final hypothesis is the following:

3. There is a quantifiable difference between data obtained from the manual PDFs and public websites which can be observed via the distance between the clusters of documents from each source.

## Datasets

### Sources and Collection Process

The focus of this project is to collect and structure data from PDF documents and websites to serve as truth data in a masked language model. Because of this, we are provided with only a small amount of data to serve as an example for the kinds of documents we will be working with. We have been given an owner's manual and a service manual for a line of PTAC systems supplied by Direct Supply to serve as an example for the structure we should expect to encounter when collecting these documents from the web. In addition, the sponsors have provided model numbers and manufacturers for 4,666 PTAC systems sold by Direct Supply that we can use to find the associated manuals that are publicly available on the internet.

- Example Owner's Manual [1]
- Example Reputable Website [2]

We are also using publicly available websites for scraping data to supplement the model, which will ideally improve model's ability to communicate using natural human language. These will need to be credible websites such as articles by companies who manufacture HVAC systems and blog posts by trustworthy authors. Given that these websites are mostly not authored from Direct Supply (and therefore could include biases), they will have been approved by the company before being used by any of our models.

Some of these biases from websites could include information that encourages people to purchase products from their website. For example, an article related to cleaning PTAC filters may recommend changing a filter several months before the service manual indicates so that they can sell more filters. This is something that will be considered when selecting websites for scraping data.

The procedure for data collection will start with a script that queries a search engine for PTAC owner's manuals, service manuals, DIY websites, HVAC/PTAC forums, and other similar web sources. Then, we will filter these sources using the provided PTAC model numbers and manufacturers as well as other heuristics we develop to limit our collected data to include only information relevant to our project. Finally, we will use PDF and HTML parsing tools to extract the relevant information and structure it for use in a masked language model. The specific structure expected by a model varies from tool to tool, but the general form is a pair of strings, one serving as a prompt that a human will ask our model and the other serving as the response to that prompt.

### Sample Sizes, Excluded Data, and Data Split

Natural language models require a massive amount of data to train from scratch. For example, OpenAI's GPT-3 model [3] was trained on approximately 500 billion words found on websites, books, and documents all across the internet. While this scale of information is out of the scope for this project, we explore the use of OpenAI's Answers endpoint [4] to create a model that uses GPT-3 to search among truth documents and generate responses to questions. The truth documents are in the form of (text, metadata) pairs of strings as JSON objects in a JSON Lines (JSONL) file.

No data is being excluded as we use our web-scraped data and PDF owner's manuals and service manuals to perform experiments with our model.

We use the troubleshooting sections from the owner's and service manuals as a sort of "FAQ" source to train a model for validation purposes. This serves as a baseline model to compare against while we collect and structure additional information for improvement. Since the exact process of using data from many different sources is still evolving, there is currently no specific training, testing, and validation split defined for this project.

## Experimental Approach

### Web Scraping

For the web scraping portion of this project, we decided to use the Beautiful Soup library [5] available for Python. This library provides an easy way to parse out the contents of a website by extracting the raw HTML and then traversing the parse tree. Using this library, we made HTTP requests to Google to obtain 80 relevant links regarding websites discussing PTAC maintenance. With these links, we made requests to the websites and extracted the HTML from them. From there, we searched common areas where important text may be found in an HTML document such as headers, paragraphs, divs, and lists. These HTML elements were searched using keywords pertaining to PTAC maintenance such as heating coils and fans to capture text discussing these aspects of a PTAC unit. This data was then saved to a CSV file so that we could later factor it into a format usable by our model.

### Owner's and Service Manual Scraping

For the PDF scraping portion of this project, we decided to use Amazon Web Services (AWS) Textract [6], which is a machine learning service that automatically extracts text, handwriting, and data from scanned documents. Before deciding upon AWS Textract, we experimented with the Python package PyPDF2. However, this strategy meant there was a considerable amount of hardcoded logic (i.e. splitting by a period, etc.). As a result, AWS provides the ability for code to be automated, even if the format of the PDFs change in the future.

### Model

Our system uses OpenAI's API, which provides access to Generative Pre-trained Transformer 3 (GPT-3), a deep language model released in June of 2020. This model was trained on approximately 500 billion words found in websites, books, and documents across the internet. OpenAI's API grants interaction with various endpoints that provide access to the GPT-3 model. Specifically, our system uses the Answers endpoint which generates responses to questions based on information contained in documents we provide. System tuning is performed through a Python script that uses OpenAI's Python API bindings. This script accesses OpenAI's servers and allows us to perform test queries, manage files containing our scraped data, list available model engines, convert text to text embeddings, and interact with any of the other API endpoints [7] not mentioned. This script is our primary means of adjusting the different aspects of our model such as engine type, data format, and hyperparameters.

### Evaluation

To evaluate the performance of our model, we use sentence vectorization and cosine similarity to provide a metric of success when comparing the output of our model to manually created answers in a validation set of question-answer pairs. This validation set was created by pulling instructions from the Troubleshooting sections in several of the PTAC owner's and service manuals. These instructions serve as answers that we might expect the model to produce when asked questions related to PTAC maintenance. Based on these answers, we manually created 67 questions that a maintenance technician or engineer may ask that could lead to those answers. The model's performance is evaluated by supplying the model with the questions from the validation set and encoding the responses using several vectorization techniques. Encodings represent a kind of meaning of text. We then calculate the cosine similarity between the model's response vectors and vectors produced by encoding answers in the validation set. A larger value of cosine similarity indicates that the model's output more closely aligns with the expected output.

### Vectorization

To visualize the data obtained from our web scraping, we first cleaned the data by removing all punctuation, numbers, and other erroneous characters present in our web data. From there, we used a count vectorizer on our dataset to prepare it for visualization using a simple bag of words model. Afterwards, we scaled our freshly vectorized dataset and then decomposed it to smaller components using Singular Value Decomposition (SVD). From these components, the two with the highest variance were used to visualize our data. Upon visualization, it was clear that using a count vectorizer by itself did not capture the needed relationships. This is because it simply counts the occurrence of words and not the actual sentiment or context behind those phrases. Other vectorizers which will be discussed later better visualize our data and give us more insight as to what good quality data looks like as opposed to poor quality data found from the internet.

We explore two techniques for sentence vectorization to see which works the best for our task. These techniques are BERT (Bi-directional Encoder Representations from Transformers) encoding and OpenAI's Embeddings endpoint. BERT, or Sentence-BERT [8] for our purposes, is a neural network that can generate embeddings that capture semantic meaning of text. OpenAI's Embeddings endpoint provides text embeddings using GPT-3's text similarity engines. We obtain these embeddings by supplying text through an API call in one of our Python scripts. Each embedding is returned as a vector of floating-point values. By implementing the vectorization portions, we can compare text together to help define a sort of "success" measure to see how well our model performs.

### Front-End Interface

The front-end interface is a simple webpage which prompts the user to enter a question to ask our model, which we named Dr. AMPS (Automated Messaging for PTAC Systems). Once the user types in their question, our model produces an output in the form of text on the screen along with text to speech so that the end user feels like they are having a conversation with a real PTAC specialist. This interface is built using simple HTML along with the React framework.

## Experimental Results

### Web Data Scraping

The web scraping aspect of this project refers to the use of Python libraries to extract text from various websites. These websites primarily consisted of blogs and manufacture websites which would then be scaped of relevant data to supply our model.

Our stakeholders proposed that we use a natural language model such as GPT-3 by including data extracted from publicly available websites. To achieve this goal, we decided to use the Beautiful Soup library for Python. This library allows for easy extraction of web data by parsing its associated HTML document via tag traversal. The tag traversal aspect of this parsing is extremely important to the success of extracting web data for one major reason: every website is structured differently. Having every website structured differently posed a significant problem for us as we could not guarantee which section of the HTML the relevant text would be located. To circumvent this issue, we decided to make a strong guess as to where the relevant text would be. After viewing a few websites manually, we decided that most of the text would be in the paragraph, div, list, and header elements of an HTML document.

With this information in mind, we made search queries using Google with the Beautiful Soup library to generate a Google search page which is the equivalent of roughly the first 100 pages of a Google search. Every link which was not a YouTube link or image link was added to a Python list and then later queried on an individual basis. When querying these links, we checked the aforementioned HTML elements for certain keywords which would frequently appear alongside PTAC maintenance tips. These keywords were filter, coil, fan, repair, and maintenance. These keywords were chosen in an admittedly unscientific way—we figured that these keywords would appear most frequently next to relevant text simply by observing some example websites. If those keywords appeared in any of the HTML elements, the contextual HTML containers that included the keywords were added to a list and then saved to be later included in the dataset. This was an easy way of gathering a large amount of data while still capturing some of the context surrounding maintenance tips. It is important to note that while this approach will give some useful data, there may also be garbage data collected as the technique used to collect the context of the repair tips was the primitive method of searching using a limited set of keywords.

After scraping these websites, we ended up with 80 links worth of data to potentially include in our dataset.

## PDF Data Scraping

We experimented with two methods of scraping data from the PDF owner's and service manuals. The first method involved using PyPDF2, a Python library for reading PDF text. The second method involved using AWS Textract, a machine learning service that uses optical character recognition to extract text from PDFs.

PyPDF2 created the need for lots of code to manually sort out the words (i.e., hardcoded rules) to extract relevant information from the manuals. This was especially challenging as PDFs differ from each other in their layout. In addition, the text returned was often corrupted or out of order. Instead, using AWS Textract simplified the process of text extraction. This service does not require the need for manually hardcoded logic to extract text, even if PDFs are formatted differently. In addition, the text returned was cleaner in both structure and accuracy of characters.

Our approach for collecting data from PDFs was to look through PTAC service/owner manuals listed on Direct Supply's website. There are about 153 different PTACs on the website, and after searching through 100 of them, we found only seven PDF manuals that were different from each other. We chose to include all seven unique PDFs in the final dataset of our system. The models/manufacturers vary across the manuals.

## Sentence Embeddings

We compare the similarity of two sentences by converting the sentences into numerical representations using sentence embeddings. Embeddings incorporate semantic meaning of sentences by converting them into vectors in an abstract embedding space. The result of the conversion is a high-dimensional floating-point vector that represents semantic meaning of a text. This vector can then be compared to another vector to assess the similarity between two texts. We experiment using two methods of vectorization: OpenAI's Embeddings and BERT Embeddings.

We created a validation set using the owner's and service manual PDFs. These PDFs contain answers to potential questions that our system may be asked, so we used the information found to generate example questions and responses. Two validation sets were used in the experiments. Validation Set A consists of 55 question-answer pairs. Validation Set B is an extension Validation Set A, containing a total of 67 question-answer pairs.

We constructed a set of human generated responses to a small list of top validation set questions. In this set, we pulled out the top five questions from the validation set that we figured might commonly show up during PTAC diagnostics. Each group member (five group members in total) manually searched through the PDF documents to generate their own answers to the questions. This set contains sentences and phrases that are semantically very similar to each other and correctly answer each of the questions chosen for this set.

Statistical tests for all experiments were conducted at a significance level of 0.05.

## Experiment I - Clustering Web Data

This section covers the first experiment ran on our web-scraped data to see which links provide high-quality data and which links provide lower-quality data.

When scraping the data, we were aware that some of the data might not be useful. This is because of not only the technique used to get the web data, but also how many Google pages deep we were going for our search results. Based solely off anecdotal evidence, we observed that after going past roughly the 4th page of a Google search result, the links start to become less and less relevant. In order to determine which links contain good data compared to bad data, we wanted to devise a way of visualizing the documents. One of the first methods we thought of was to use a bag of words model to visualize our web text data. We hypothesized that the first 10 links would yield good quality data and would be clustered closer together than links obtained after that. Additionally, we thought that we would be able to determine extremely poor data by setting some sort of distance threshold for how far off the data is from the good data. If a data point is beyond the threshold point, then we could remove it so that we would not corrupt our data pool with garbage. The results of this visualization are shown in Figure 1.

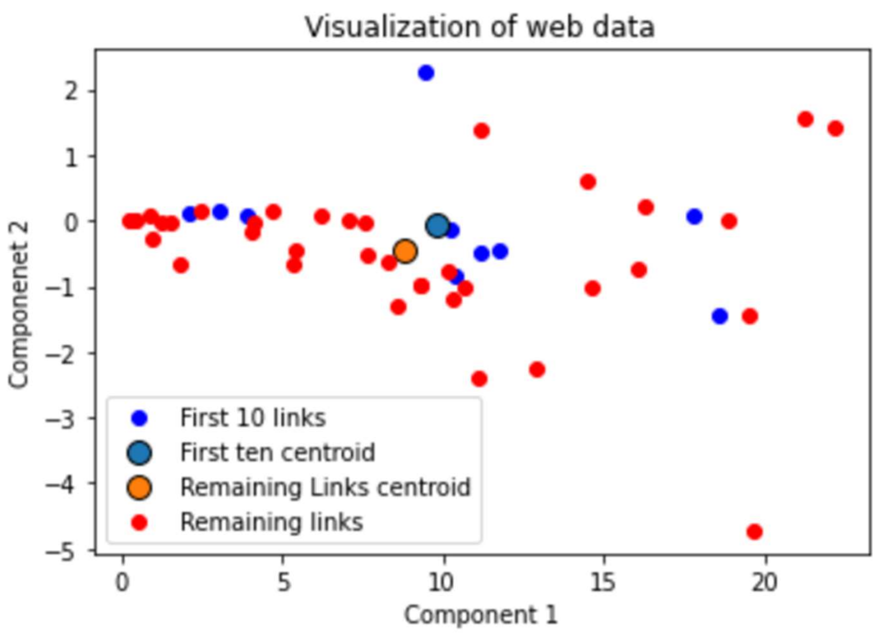

*Figure 1 - Decomposed Bag-of-Words Representation of Webpages Scraped from Google*

From Figure 1, the centroids of the first ten links and the remaining links are very close together in this space. In addition, the points of both groups are evenly distributed throughout this space without any clear separation. This indicates that using a bag-of-words model is unlikely to grant us the ability to distinguish between higher and lower quality data sources.

## Experiment II - Comparing Performance Using Cosine Similarity and Euclidean Distance

To get an idea of how similarity looks like in a variety of settings, we performed similarity measures among different groups using both cosine similarity and Euclidean distance.

The first control group tested was the mean pairwise validation set answers and model outputs. For this set, we queried our model to receive the output for every question in the validation set and calculated the pairwise similarities between the validation set answers and actual outputs. This gives a sense for how well the model output matches the expected output.

The second control group tested was the mean quintuple-wise human generated responses to the top-5 validation set questions. This group gives a sense for what a similarity metric value might look like when comparing between truly similar sentences.

The third control group tested was the validation set answers. In this group, we calculate the mean similarity between every possible pair of sentences in this set. This gives a sense of scale of the similarity measure in the PTAC domain between sentences that are different from each other in meaning.

The model used in this experiment was our minimum viable model with our initial dataset of PDF data and web-scraped data with documents truncated if they contained too many tokens for OpenAI's servers to process. The model used the Ada search engine and the Ada output generation engine.

We calculated this experiment using both cosine similarity and Euclidean distance. The pairwise similarities between the validation set answers and model outputs using cosine similarity and Euclidean distance with OpenAI's Embeddings are plotted in Figure 2 below.

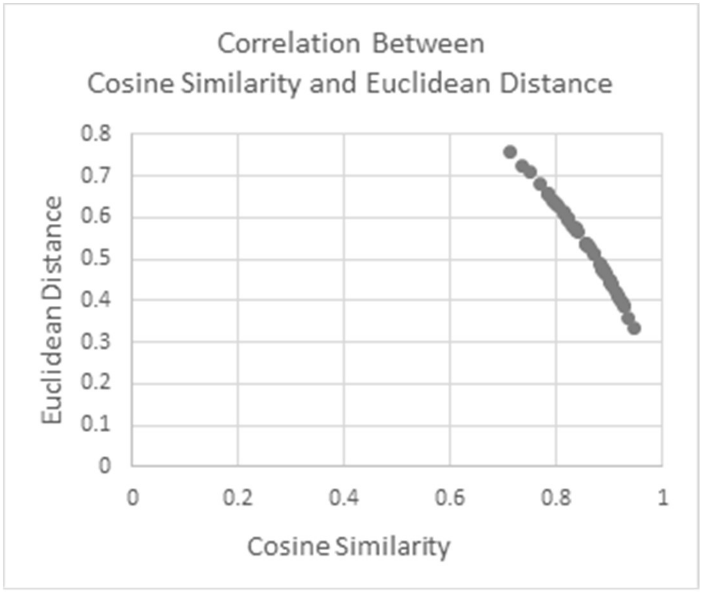

*Figure 2 - Correlation Between Cosine Similarity and Euclidean Distance in OpenAI's Embeddings Space*

Figure 2 shows that there is a strong correlation between cosine similarity and Euclidean distance in the embedding space. This is confirmed using a Spearman's Rank test which yielded a p-value < 0.001. This indicates the existence of a transformation between cosine similarity and Euclidean distance, concluding that both metrics are equally informative. Since cosine similarity has a standardized range of [-1, 1], we will use cosine similarity as our similarity metric for the remainder of the experiments.

We calculated the mean similarity measures for the three control groups (pairwise validation answers and outputs, quintuple-wise human answers, and permutation validation answers). This was performed using both OpenAI's embeddings and BERT embeddings. The validation set used for this experiment was Validation Set A which contains 55 question-response pairs.

Figure 3 shows the mean cosine similarities of these four groups using OpenAI's embeddings with the Ada engine.

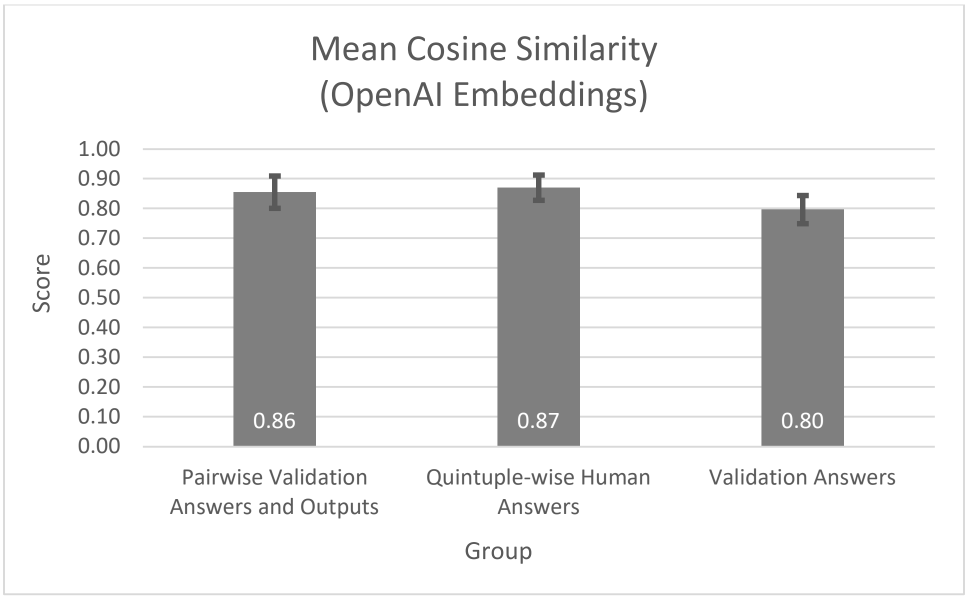

*Figure 3 - Mean Cosine Similarity Measures Between Four Control Groups Using OpenAI's Embeddings. Error bars show one standard deviation above and one standard deviation below the mean.*

The mean quintuple-wise similarity between the human generated answers to the top five validation set questions had the highest similarity measure of 0.87 in OpenAI's embedding space. This was followed by the pairwise validation set answers and outputs, then the validation set model outputs, and finally the validation set answers. The similarity measures here all belonged to a small range of 0.82 to 0.87.

We performed an ANOVA statistical test to test if the distributions of the control groups are different using OpenAI embeddings. This test yielded a p-value < 0.001, indicating statistical significance and concluding that the cosine similarity distributions between the control groups are different. In addition, we performed a post-hoc ANOVA series with a Bonferroni correction to find out which distributions are different from each other. The results are summarized in Table 1 below.

| **Post-Hoc Test** | **Bonferroni Adjusted P-value** |
|---|---:|
| Pairwise vs Quintuple-wise | 1.000 |
| Pairwise vs Validation | < 0.001 |
| Quintuple-wise vs Validation | 0.002 |

*Table 1 - Summary of Post-Hoc ANOVA Test Results for Experiment II (OpenAI Embeddings)*

The permutation validation answers control group distribution was found to be statistically different than both the other control group distributions.

Figure 4 below shows the mean cosine similarities of the four control groups using BERT embeddings with the "all-mpnet-base-v2" model.

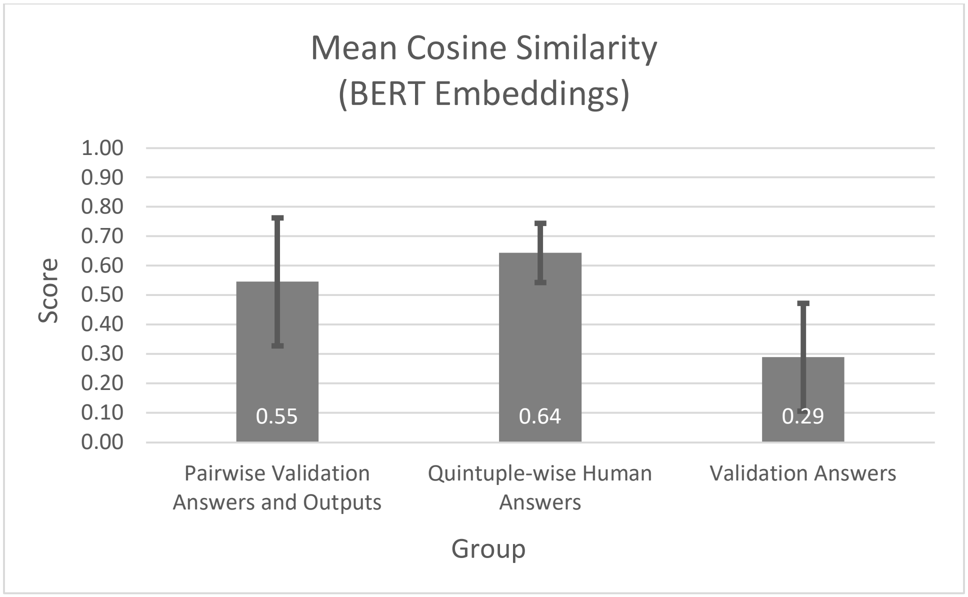

*Figure 4 - Mean Cosine Similarity Measures Between Four Control Groups Using BERT Embeddings. Error bars show one standard deviation above and one standard deviation below the mean.*

In the BERT embedding space, the mean quintuple-wise similarity of human answers again had the highest similarity measure, this time of 0.64. This was followed by the pairwise validation answers and outputs, the validation set model outputs, and the validation set answers. The similarity values here belonged to a wider range of 0.28 to 0.64.

We performed an ANOVA statistical test to test if the distributions of the control groups are different using BERT embeddings. This test yielded a p-value < 0.001, indicating statistical significance and concluding that the cosine similarity distributions between the control groups are different. In addition, we performed another post-hoc ANOVA series to see which distributions are different from each other. The results are summarized in Table 2 below.

| **Post-Hoc Test** | **Bonferroni Adjusted P-value** |
|---|---:|
| Pairwise vs Quintuple-wise | 0.975 |
| Pairwise vs Validation | < 0.001 |
| Quintuple-wise vs Validation | < 0.001 |

*Table 2 - Summary of Post-Hoc ANOVA Test Results for Experiment II (OpenAI Embeddings)*

The permutation validation answers control group distribution was again found to be statistically different than both the other control group distributions.

## Experiment III - Comparing Models Created with Web-Only Dataset, PDF-Only Dataset, and Combined Web and PDF Dataset

This experiment aimed to compare the performance of models created using different sources of data. Specifically, we compared performance of models to the validation set using web-only data, PDF-only data, and a combined set of web and PDF data.

This experiment was performed by creating three datasets. The first dataset of PDF-only data was collected using AWS Textract on one service manual PDF. The second dataset of web-only data contained the same web-scraped data that was collected in experiment I, which was also included in our minimum viable model. The third dataset of combined data was a concatenation of the first and second datasets.

The validation set used for this experiment was Validation Set B which contains 67 question-response pairs. The model used the Ada search engine and the Curie output engine. The embeddings were generated using the BERT "all-mpnet-base-v2" model.

Figure 5 shows the mean pairwise cosine similarity measures between validation set answers and model outputs in BERT embedding space for models using each dataset.

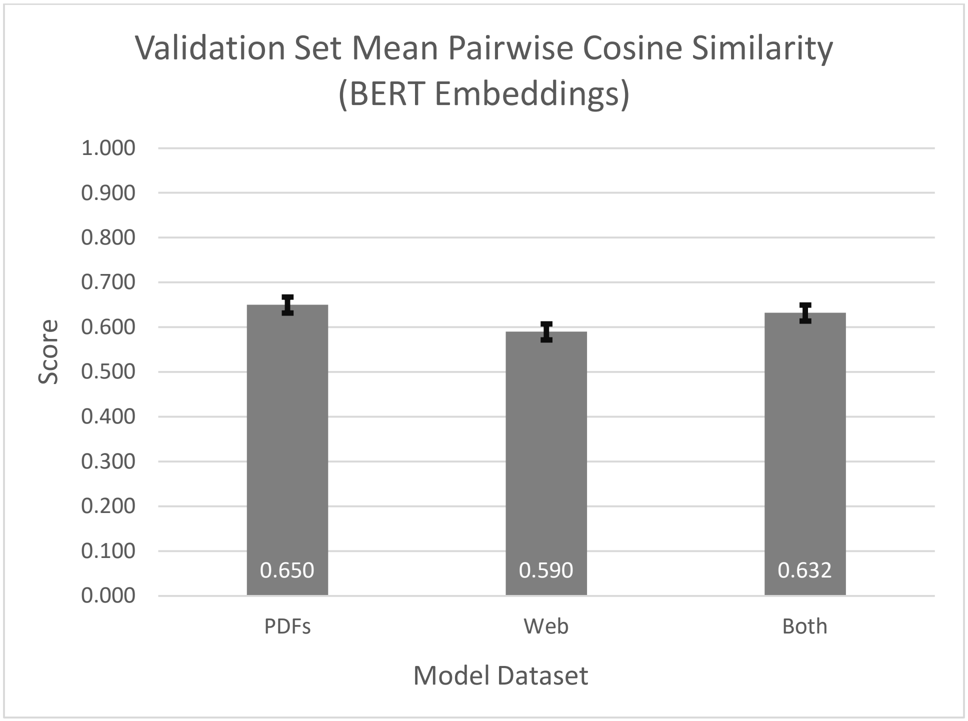

*Figure 5 - Mean Pairwise Similarities Between Validation Set Answers and Model Outputs Across Models Using PDF-Only, Web-Only, and Combined Data (BERT Embeddings).*
*Error bars show one standard deviation above and one standard deviation below the mean.*

The model using PDF-only data produced the highest mean pairwise similarity with a score of 0.650 followed by the combined dataset score of 0.632 and then by the web-only score of 0.590. The difference between the highest and lowest mean similarity for these datasets is about 0.06.

We performed an ANOVA test to determine if there is a difference between the distributions of the control groups tested. This test resulted in a p-value of 0.002, indicating that the distributions are statistically different. We performed another post-hoc ANOVA series to see which of these distributions are different from each other. The results are summarized in Table 3.

| Post-Hoc Test | Bonferroni Adjusted P-value |
|---|---|
| PDF vs Web | 0.001 |
| PDF vs Both | 0.856 |
| Web vs Both | 0.040 |

*Table 3 - Summary of Post-Hoc ANOVA Test Results for Experiment III (BERT Embeddings)*

The web dataset distribution was found to be statistically different from both the PDF-only dataset distribution and the combined PDF-and-web dataset distribution.

## Experiment IV - Comparing Models with Different Engine Tiers

This final experiment aimed to assess the performance of the model when using different output engines. We validated four models trained on different output engine tiers: Ada, Babbage, Curie, and Davinci. The tiers of engines vary in terms of ability and price, with Ada being the cheapest and least capable model and Davinci being the most expensive and most capable model. The Answers task in OpenAI requires two model engines: one to search and rank documents to find where the answers are located and one to generate the responses based on the information found. This experiment held constant the search engine (Ada) and explored the performance of the different model output engines.

The dataset used in this experiment was a PDF-only dataset collected using AWS Textract on seven unique owner's and service manuals. The validation set used for this experiment was Validation Set B, which contains 67 question-response pairs. The embeddings were generated using the BERT "all-mpnet-base-v2" model.

Figure 6 below shows the mean pairwise cosine similarity of validation set answers and model outputs across each model engine.

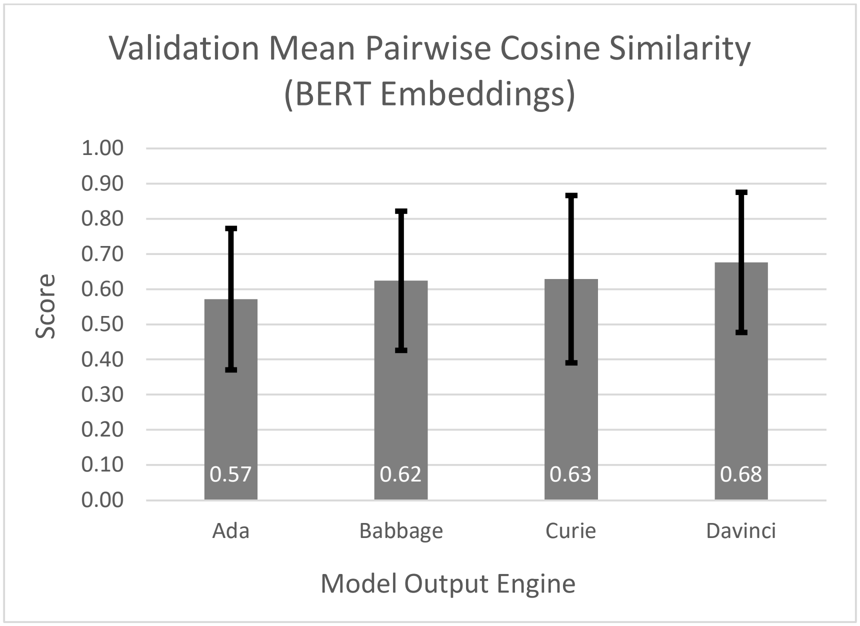

*Figure 6 - Mean Pairwise Similarities Between Validation Set Answers and Model Outputs Across Model Engine Tiers. Error bars show one standard deviation above and one standard deviation below the mean.*

As the capability and price of the engine increases, so does the similarity score of the model outputs to the validation set's expected outputs. The cheapest and least capable model, Ada, produced a mean similarity score of 0.57. The most expensive and most capable model, Davinci, produced a mean similarity score of 0.68. Babbage and Curie, respectively, produced scores of 0.62 and 0.63.

We performed an ANOVA statistical test to test if the model engine truly affects the similarity of the output to the validation set. This test yielded a p-value < 0.001, indicating statistical significance and concluding that the cosine similarity distributions between the control groups are different. In addition, we performed a post-hoc ANOVA series to see which distributions are different from each other. The results are summarized in Table 4 below.

| Post-Hoc Test | Bonferroni Adjusted P-value |
|---|---:|
| Ada vs Babbage | 0.022 |
| Ada vs Curie | 0.009 |
| Ada vs Davinci | < 0.001 |
| Babbage vs Curie | 1.000 |
| Babbage vs Davinci | 0.021 |
| Curie vs Davinci | 0.045 |

*Table 4 - Summary of Post-Hoc ANOVA Test Results for Experiment IV*

From Table 4, the only post-hoc test that did not indicate statistical significance was the test between the distributions of the Babbage and Curie engines.

## Ethical Considerations

When evaluating ethical considerations within our project, two aspects stand out. The model could inform the user to replace parts more often than needed, and it could also use toxic language when responding to the user.

The model informing the user to replace parts more often than needed causes concerns for several reasons. One is that the end user would unnecessarily spend more money. A result of this could be legal action against Direct Supply related to false advertising or misinformation. In addition, the customer's finances would be negatively affected, meaning that Direct Supply's professional relationships could deteriorate, and this could even result in lost customers. Steps taken to minimize this concern are quite challenging in the current state of the model. GPT-3 does give a confidence score based on how well it thinks the documents answer each prompted question. However, even if the result it feeds back is wrong (i.e. recommending the replacement of the air filter every week instead of every 3 months), the confidence score from GPT-3 could still be very high. Even if we manually compare that answer to our validation set, the metrics could still be acceptable because the answer's verbiage is still PTAC-related. One article [9] related to this issue describes a lawsuit against Match Group Inc. (a dating app corporation), where they were sued for $844 million for failing to screen out fake accounts and messages. Match won the lawsuit, and a mitigation they claimed they had was that they blocked 96% of bots and fake accounts from their site. For our project, one solution for this misinformation concern is to filter out incorrect information, but determining what responses are correct and incorrect is still a work in progress.

The other main ethical concern in our model is that it could generate toxic language in response to questions prompted by a user. This could have significant implications regarding the professional image of Direct Supply, especially if the responses go so far as to offend or insult the user. While this is not likely to happen in the typical use case, an open-ended, creative model such as GPT-3 always has the possibility of perhaps being a bit too creative with its responses. One notable example for a model exhibiting toxic behavior is Microsoft's chatbot, Tay, that was tested in 2016. This chatbot [10] was raided by internet trolls who exploited a "repeat after me" feature present in the model by feeding it many toxic sentences and phrases. The model learned this language and began using it in responses to other questions, many of which were completely unrelated to how Tay responded. Microsoft responded by abandoning Tay and developing a politically correct chatbot designed to shut down conversation about sensitive topics. Microsoft ultimately found that the issue of toxic language is largely societal in nature and is not easily solved. OpenAI recognizes this concern and provides an API endpoint [11] specifically for the use of detecting toxic language output by applications built using GPT-3. An application can query the API and supply with it the output of the model, and the endpoint will return a value of 0, 1, or 2 based on the type of language it finds. A value of 0 indicates that the text is safe. A value of 1 indicates that the response contains text that may involve sensitive topics such as politics or religion. A value of 2 indicates that the text is unsafe, meaning that it contains language deemed as toxic, such as swear words or bigotry. This filter would be easily incorporated into our system, though it may increase the latency between questions and responses. However, that is a small price to pay for avoiding potential negative consequences that may arise from toxic responses generated by our model.

## Discussion and Conclusions

This section discusses the results concluded by our experiments. AWS Textract was found to be superior to PyPDF2 for scraping PDF documents to use in our model. Clustering web documents using a bag-of-words representation does not provide a nice separation of document points of which humans can manually distinguish the quality. BERT embeddings give a wider range of representation for sentences than OpenAI's embeddings, and we introduce a validation strategy that involves computing the mean pairwise cosine similarity between expected answers and model outputs. The PDF owner's and service manuals produce the best results for our model in terms of our validation strategy compared to using web data or a combined web-and-PDF dataset. Finally, we find that model quality scales with engine cost, and we recommend using Ada search with Curie model output generation.

### PDF Scraping

After performing experiments with PyPDF2 and AWS Textract, AWS Textract ended up as the much better option. With PyPDF2, extracting text involved lots of manual, hard-coded logic that depended on the format of each PDF. However, AWS Textract was much more automated, which is important because many PDFs are formatted differently from each other. This will also make it easier to grab text from PDFs in the future when manufacturers publish manuals for new models of PTACs.

### Experiment I - Clustering Web Data and PDF Data

Looking at Figure 1 in the experimental results section, one can see that our visualization efforts did not exactly yield fruitful results. In order to confirm that there was nothing useful distinguishing the two groups, we ran a statistical test to determine significance. More specifically, we found the average values for all points in the two groups and plotted them as centroids. We then took the distances between every point in each group and their respective centroid and ran a t test to determine if there is a statically significant difference in the mean distances of each group. If there is a statistically significant difference, then we can say there is some sort of relationship being captured between the two groups. After running the t test, we received a p-value of 0.45 which is well above the 0.05 threshold. Therefore, we determined that there is no statistically significant difference between the average distances of the two groups and their respective centroid. This can be attributed to the fact that we are using a simple bag-of-words model to vectorize the documents. Since this model simply represents the frequency of words and not the semantics and context, this method of analyzing our data will not be sufficient for validating our data. However, this was an important first step in attempting to quantify textual information.

While this attempt at visualizing our data in order to make useful observations regarding the quality of our data seemed like a good idea initially, it became obvious as to why it did not lead to any fruitful results after the fact. Despite the lack of any sort of qualitative results to assess our data with, this was still a useful experiment to run. The usefulness of this experiment was not what the data did show, rather, it was what the data did not show. Since this experiment was the most naïve approach to visualize our data in order to judge its quality, it was worthwhile to conduct it to clearly illustrate the shortcomings of visualizing our data in this manner. This experiment's lack of quality results was what prompted the need to further explore our data using different vectorizers in order to capture the semantical context surrounding keywords rather than simply the frequency of those keywords alone.

## Experiment II - Comparing Performance Using Cosine Similarity and Euclidean Distance

Comparing the results in Figure 4 to the results in Figure 3, the BERT embeddings show more promise than the OpenAI embeddings in capturing the actual semantic meaning of sentences in this problem domain. Though both embedding strategies resulted in distributions with similar statistical differences, the range of similarities is much larger when using BERT embeddings. The OpenAI strategy results in distributions with similar means and low variance, whereas the BERT strategy results in distributions that appear to have a wider range of means and higher variance. In Figure 4, the first two groups, those calculated using pairwise or quintuple-wise similarities between sentences that should be semantically similar, show much higher similarities than the last group, which is calculated using permutation similarities among sentences with higher semantic variation. We then hypothesize that the BERT model can produce a more elaborate representation of the semantic meaning of sentences based on this initial experiment. This can be tested through further experimentation by hiring a subject matter expert to rank responses of the model among categories such as correctness, tone, and generalizability to assess if the rankings agree with the cosine similarity scores.

Regardless of the embedding strategy used, both post-hoc tests indicated that the permutation validation answers had a significantly different distribution than both the pairwise validation answer-outputs and the quintuple-wise human generated answers. Figure 4 shows that the highest similarity was between human generated responses to the same validation questions. This makes sense because the sentences for each question are varied in wording and are not varied in meaning. This also supports that this validation strategy can be useful in quantitatively assessing the performance of the model. The pairwise similarity between validation set answers and model outputs had the next highest similarity. This indicates that this model is at least somewhat intelligently understanding the questions and generating appropriate responses. The last control, which involved calculating the permutation similarity among the validation set answers, had considerably a lower mean similarity, which is expected given that these sentences vary in wording and semantic meaning more strongly than the other groups. There is still some sense of similarity due to the problem domain as the model uses PTAC information to generate responses.

## Experiment III - Comparing Models Created with Web-Only Dataset, PDF-Only Dataset, and Combined Web and PDF Dataset

The post-hoc ANOVA tests conclude that the model trained using web-only data had a different similarity distribution than the other two models. Figure 5 shows that the PDF-only dataset produced the highest mean pairwise similarity of all the datasets. This makes sense because the true answers in the validation set were generated using PDF owner's and service manuals. The output most similar to the true validation set answers comes from the model that only used PDFs to generate responses. Following this, the model with the combined web and PDF data produced the next-highest similarity to the validation set, and the model with the web-only data produced the lowest similarity to the validation set.

This experiment does not conclude that a model created using PDF-only data is more correct than a model created using some or all web-scraped data. It simply shows that the model's output is most similar to the output expected by the validation set when generating responses using PDF data. The web-scraped data, for instance, might allow the model to generalize better to a wider range of problems, or it might allow the model to use more understandable language than that found in PDF service manuals. This, however, is not easy to quantitatively assess. Future experimentation using a subject matter expert to qualitatively assess the model's output may provide better insight to the realistic quality and accuracy of the model expected by end users.

## Experiment IV - Comparing Models with Different Engine Tiers

Figure 6 shows that as the price and capability of the engine increases, the mean pairwise similarity of model outputs to the validation set answers also increases. This is as expected given that the more expensive and more capable options produce higher quality results. There is thus a tradeoff here between the expense of running this model and the quality of the answers produced by it. The total cost of the final three experiments was $38.18.

The Answers system relies on two engines: one to search through documents and one to generate responses. As more documents are used in the model (i.e., more data is considered), the cost of the searching step increases. In this experiment, with 7 PDFs split across 90 JSON objects, the cost of search using the Ada engine averaged about $0.03-0.08 per question. The cost to generate the output for the Ada, Babbage, Curie, and Davinci engines, respectively, averaged about $0.001, $0.002, $0.01, and $0.11 per question. Given that search is a significant portion of the expense, it is recommended to use the cheapest model, Ada, for this task. It may also make sense to scale back the output generation engine to Curie to avoid an extra ten cents per question using Davinci. Though the post-hoc test found no statistical difference between the distributions of Babbage and Curie and the quantitative similarity of Ada or Babbage does not appear much lower than the similarity of Curie or Davinci, using Ada or Babbage for model output produces rather weak responses qualitatively. For example, the responses of these models tend to repeat over and over (e.g., one of these outputs of the Ada model during the experiment was "The fan is not working. The fan is not working. The fan is not working…" with this phrase repeating until the maximum token limit was reached). Therefore, it is not recommended to use Ada or Babbage for model output generation.

## Future Work

After the completion of this project, it is our recommendation that one should use more sophisticated means of collecting data from internet sources. The methods used to obtain this data were not strictly implemented with the concern of making the best possible choice due to our eleven-week deadline and our relative inexperience in data collection pipelines. As a result of this, our simple collection method may have allowed poor quality data to seep into our web dataset and, even more likely, miss high quality data that was either not in the HTML elements we considered or did not show up in our searches on Google. To rectify this issue, we recommend that anyone picking this project up in the future should investigate more robust ways of collecting data from web sources. Ideally, websites should have some sort of credibility rank, possibly determined by a subject matter expert. It could also be worth experimenting with different splits of PDF data and web data, as an increase in the quality of web data could improve the performance of the model when using a combined dataset.

Additionally, more work still needs to be done on the validation side of things. While our model can produce seemingly correct results based on certain questions, we cannot confidently confirm that they are correct. This issue can be reduced by using a PTAC subject matter expert to screen certain answers from our model to determine if they are correct in the more ambiguous cases. We also recommend that a subject matter expert reviews and edits or recreates the validation dataset used throughout this project to reflect real-world scenarios more accurately. Finally, it is recommended to further experiment with different validation strategies in general, where the subject matter expert could assist in ranking model outputs in addition to revising the validation set itself.

## Main Takeaways

This project involved implementing an AI chatbot to be used by maintenance engineers and technicians to answer PTAC diagnostic questions regarding common maintenance problems. We explored several topics including scraping websites and PDF manuals for training data, developing a validation strategy, experimenting with vectorization techniques, and implementing an AI chatbot using the recently developed GPT-3 natural language model from OpenAI. From our experiments, we established a capable validation strategy based on calculating the mean pairwise cosine similarity between a set of expected outputs of example questions and the model's actual outputs to those questions. This validation strategy makes use of the BERT model to generate numerical representations of sentences called sentence embeddings which can be compared using cosine similarity. Using this validation strategy, we find that the highest mean cosine similarity for the model can be produced by using a PDF-only dataset, the Ada search engine, and the Davinci model output generation engine. For a better balance of performance and cost, the Curie engine should be used instead of the Davinci engine for model output generation.

The repository storing the scripts and documents used for this project can be accessed by the following link: https://github.com/RhodesKeegan/DirectSupplyProject/

## Bibliography

### Tools, APIs, and Example Documents

[1] Sample PDF Owner’s Manual

- ➢ https://github.com/RhodesKeegan/DirectSupplyProject/blob/main/PDF%20Scraping/PDFs/sample%201.pdf

[2] Sample Website for Data Scraping

- ➢ https://store.directsupply.com/Promotions/PTAC-preventive-maintenance-checklist.html

[4] OpenAI Answers Guide

- ➢ https://beta.openai.com/docs/guides/answers

[5] Beautiful Soup Documentation

- ➢ https://beautiful-soup-4.readthedocs.io/en/latest/

[6] AWS Textract Homepage

- ➢ https://aws.amazon.com/textract/

[7] OpenAI API Overview

- ➢ https://beta.openai.com/docs/api-reference/introduction

[11] OpenAI Content Filter Explanation

- ➢ https://beta.openai.com/docs/engines/content-filter

### Papers and News Articles

[3] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language Models are Few-Shot Learners. arXiv preprint arXiv:2005.14165. Retrieved May 13, 2022, from https://arxiv.org/abs/2005.14165

[8] Reimers, N., & Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. arXiv preprint arXiv:1908.10084. Retrieved April 16, 2022, from https://arxiv.org/abs/1908.10084

[9] Calkins, L. B. (2022, March 24). Match Group avoids $844 million in FTC claims on judge's ruling. BNN Bloomberg. Retrieved April 25, 2022, from https://www.bnnbloomberg.ca/match-group-avoids-844-million-in-ftc-claims-on-judge-s-ruling-1.1742789

[10] Schwartz, O. (2019, November 25). In 2016, Microsoft's racist chatbot revealed the dangers of online conversation. IEEE Spectrum. Retrieved April 26, 2022, from https://spectrum.ieee.org/in-2016-microsofts-racist-chatbot-revealed-the-dangers-of-online-conversation

## Appendix

### Experiment III - Comparing Models Created with Web-Only Dataset, PDF-Only Dataset, and Combined Web and PDF Dataset

We also compared the embedding strategies of OpenAI and BERT again, this time where the OpenAI embeddings were calculated using the Davinci engine and the BERT embeddings were again calculated using the "all-mpnet-base-v2" model.

Figure 7 shows the mean pairwise cosine similarity measures between validation set answers and model outputs in OpenAI's embedding space for models using each dataset.

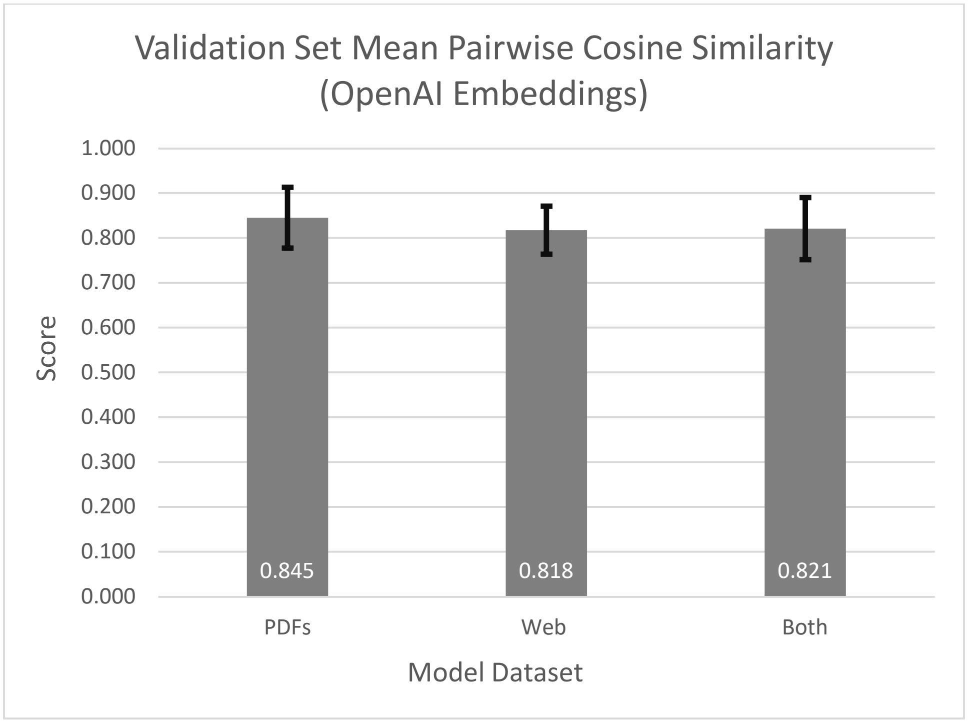

*Figure 7 - Mean Pairwise Similarities Between Validation Set Answers and Model Outputs Across Models Using PDF-Only, Web-Only, and Combined Data (OpenAI Embeddings).*
*Error bars show one standard deviation above and one standard deviation below the mean.*

The model using PDF-only data produced the highest mean pairwise similarity of 0.845 followed by the combined dataset score of 0.821 and then by the web-only score of 0.818. The range of similarities here in the OpenAI embedding space is low, with scores varying only by 0.024.

We performed an ANOVA statistical test to test if the distributions of the control groups are different using OpenAI embeddings. This test yielded a p-value < 0.001, indicating statistical significance and concluding that the cosine similarity distributions between the different datasets are different. In addition, we performed another post-hoc ANOVA series to see which distributions are different from each other. The results are summarized in Table 1.

| Post-Hoc Test | Bonferroni Adjusted P-value |
|---|---|
| PDF vs Web | < 0.001 |
| PDF vs Both | 0.003 |
| Web vs Both | 1.000 |

*Table 5 - Summary of Post-Hoc ANOVA Test Results for Experiment III (OpenAI Embeddings)*

Comparing the results in Figure 5 to the results in Figure 7, the BERT embeddings again show more promise in capturing the semantic meaning of sentences. The post-hoc test results for the OpenAI embeddings were different than the results for the BERT embeddings. The results for the OpenAI embeddings indicated that the PDF distribution was different from the other distributions, where the results for the BERT embeddings indicated that the web distribution was different from the other distributions. However, both strategies resulted in the same ordering of mean similarities, where the PDF distribution had the highest mean similarity and the web distribution had the lowest mean similarity. The range of similarities is again larger when using BERT embeddings over OpenAI's embeddings, so it seems that OpenAI's embeddings cannot distinguish semantic meaning with as much detail. Therefore, we chose to use embeddings generated from the BERT model in Experiment IV.

## Experiment IV - Comparing Models with Different Engine Tiers

Table 6 shows some example model outputs when asked questions from the validation set. This model uses the same the configuration that was used in Experiment IV with the Curie output generation engine.

| Validation Question | Validation Answer | Model Output |
|---|---|---|
| What should I use to clean the PTAC? | To clean, use water and a mild detergent | Use a soft cloth dampened with a mild detergent solution. Do not use any type of abrasive or solvent. |
| How do I remove the front panel? | Pull out at the bottom to release it from the tabs and then lift up | The front panel is removed by pulling out at the bottom to release it, then lifting it up to clear the rail along the chassis top. |
| Why is the PTAC spitting out water? | The position might need to be adjusted. Make sure the PTAC is level | The PTAC is designed to operate in a dry environment. If the unit is installed in a wet environment, the unit will spit out water. |
| What is the maximum temperature that I can set? | The maximum temperature is eighty four degrees | The maximum temperature that can be set is 84°F/29°C. |
| What kind of refrigerant is used? | The PTAC unit uses R-410A refrigerant | The Friedrich PTAC uses R-410A refrigerant. |
| How do I clean the air filters? | Pull up on the air filter to remove, vacuum off heavy soil, run water through the filter, and dry thoroughly | The air filters are interchangeable and will fit in immediately. Either the right or left side. |

*Table 6 - Example Model Outputs Using Curie Output Engine*

The model performs best when the answer is simple and straightforward, such as the type of refrigerant used by the system or the maximum temperature that can be set. When the answer is more open-ended or requires several sentences of explanation, the model usually gives an answer that is at least partially correct or uses similar language included in the question. This sample output emphasizes the difficulty of automatically validating the output of our model. Some of the responses appear correct, and most responses use correct grammar, but there is not a definitive way to evaluate the quality or correctness of responses. Using a subject matter expert to rank responses may prove to be a better strategy for validating the output of the model.