Feynn Labs Internship

(January, 2023)

Project Report on

AI Product/Service Prototyping for Small Businesses

Submitted By,

Arya Shah

Batch-2-23-SB MLI

Topic Chosen:

AI In IT Operations for Service Helpdesk Automation

Table of Contents

Introduction

Any organization that uses technology to run their business will have an IT organization. And because they are using technology on a large scale, they'll also have a department for IT Operations, or ITOps, for short.

ITOps refers to a set of processes and services that are managed by the IT organization to deliver technology services. ITOps typically serves internal employees, as well as external customers who use their organization's services. But what are some of the key functions of ITOps? Well, essentially, it is the planning, deployment, and maintenance of the organizations hardware and software assets. They set up laptops, install business software, and deploy networks within an office or at a data center. They do this both for technology that is deployed on individual clients like laptops, as well as services that are deployed in a data cluster.

Problem Statement

An IT help desk provides a centralized resource to answer questions and troubleshoot problems and AI can improve service levels and reduce costs. Hence, the problem statement is to build a natural language model that can take a new user question and find the closest question in the dataset. Then, we take the corresponding FAQ article and return it to the user.

Market/Customer Need Assessment

Typically, an employee or a customer has trouble using the products or services. Say a user is unable to print a Word document. They reach out to a centralized IT help desk either through a phone, email, or chat. Someone from the IT help desk answers questions and help the user resolve his issues. Usually, most user questions are simple, have been experienced in the past, and have ready-made answers. These answers are usually documented in guides or frequently asked questions documents. While answers are easy to find, human assistance is needed to understand the user's questions in their natural language.

Alternatives like self-help websites require the user to go through a number of, yes/no type questions to find the answers. AI can help here. It can be used to understand the questions asked by the user in natural language and narrow it down to the relevant FAQ articles.

Target Specifications and Characterization

Automating the concept of solving problems addressed in FAQs

With the growing IT services sector, a lot of customer queries and helplines are set up but not all calls can be addressed and most of the questions that come in are of FAQ in nature.

Wide range of questions and grievances with similar solutions

Same problem can be phrased in n-number of ways by different people. Having a smart system which can understand and appropriately provide a solution without wasting time is beneficial

External Search and Information Sources

The data set is available in the file helpdesk_dataset.csv.

Question	LinkToAnswer
question	link to answer
10 unique values	http://faq/mac-do... 30% http://faq/mac-bo... 30% Other (4) 40%
My Mac does not boot, what can I do ?	http://faq/mac-does-not-boot
Can Mac Air get infected by a Virus	http://faq/mac-book-virus
My Mac is having boot problems, how do I fix it?	http://faq/mac-does-not-boot
Do I need an anti virus on my Mac?	http://faq/mac-book-virus
I have trouble connecting my monitor to my Mac. Can you please help?	http://faq/mac-monitor-setup
When my Mac boots, it shows an unsupporterd software error	http://faq/mac-does-not-boot

The CSV has two columns.

The first column called Question contains a natural language question that a user would ask.

The second column, LinkToAnswer, contains a link to an FAQ article that provides answers to this question.

Important to note that this is a really small dataset created for demonstration purposes. In real use cases, there would be tens of questions that would point to the same FAQ. The same question may be phrased in multiple ways in order to help the model learn multiple ways in which the question can be asked.

It is highly recommended to have an elaborate dataset for accurate results.

Applicable Patents

The following Patents find similarity with the current concept and are applicable:

1. Bihani, A. (2018). FAQtor : Automatic FAQ generation using online forums ∗.
2. E. Sneiders, "Automated FAQ answering with question-specific knowledge representation for web self-service," 2009 2nd Conference on Human System Interactions, Catania, Italy, 2009, pp. 298-305, doi: 10.1109/HSI.2009.5090996.
3. https://thebotplatform.com/solutions/hr-people/helpdesk-and-faqs/

The first two pointers are research articles written by authors aiming to address this problem statement. The third pointer is actually a service provider for automated helpdesk solutions and FAQ answering.

Applicable Regulations

The following regulations are applicable:

1. Laws and Regulations for Data Collection: Some websites might have a policy against collecting customer data in form of reviews and ratings.
2. Laws and Regulations for Scraping Web Data: It is important to protect the privacy and intention with which the data was extracted.

3. Correctness of Information: Due diligence and compliance to be needed for correct instructions provided by the system while recommending FAQ Solutions.

Applicable Constraints

Following constraints seem applicable:

1. Constraint of domain and selection of most relevant domains
2. Complying with regulations related to privacy and anonymity
3. Gathering huge amount of question answer pairs for selected domain

Business Model and Monetization Opportunities

Following are the business model and monetization opportunities:

1. Providing API access with huge knowledge base of customer FAQs solutions
2. Providing custom FAQ solutions for selected domains

By providing API access, we can charge a standard amount for certain number of API requests.

After creating a huge knowledge base of FAQ solutions, we can either sell the knowledge base to other organization on subscription or create custom end-to-end solutions for them.

Businesses that require excessive customer care or troubleshooting will benefit the most from utilizing this concept.

Concept Generation

Let's review the functions of an IT help desk and how AI can improve service levels and reduce costs. An IT help desk provides a centralized resource to answer questions and troubleshoot problems.

Typically, an employee or a customer has trouble using the products or services. Say a user is unable to print a Word document. They reach out to a centralized IT help desk either through a phone, email, or chat. Someone from the IT help

desk answers questions and help the user resolve his issues. Usually, most user questions are simple, have been experienced in the past, and have ready-made answers. These answers are usually documented in guides or frequently asked questions documents. While answers are easy to find, human assistance is needed to understand the user's questions in their natural language.

Alternatives like self-help websites require the user to go through a number of, yes/no type questions to find the answers. AI can help here. It can be used to understand the questions asked by the user in natural language and narrow it down to the relevant FAQ articles. In the subsequent sections, I will demonstrate such a use case.

We will take a very simple approach for this use case for demonstration purposes. We will use a data set that contains a list of FAQ articles and a corresponding set of questions. The FAQ articles can have multiple questions associated with them. We will then build a natural language model that can take a new user question and find the closest question in the dataset. Then, we take the corresponding FAQ article and return it to the user. Next, we will review the techniques that we will be using to build the model.

Concept Development

1. Latent Semantic Analysis (LSA) and Latent Semantic Indexing (LSI)

I will review two of the major techniques used for building the self-service help desk. Machine learning algorithms work only with numeric data. They do not understand text.

One of the most recent and popular techniques to convert text into its numeric representation is called Latent Semantic Analysis or LSA. It can use the vectorized representation of documents to analyze relationships and arrive at a similarity model.

It builds an index using the latent semantic indexing, or LSI technique, which measures the relationships between terms in an unstructured collection of text. The index can then be used to find similar documents based on commonly occurring phrases between the documents.

2. Building A Document Vector

We will convert the questions in the data set into a document vector for ingestion by LSI models.

First, we ensure that all the package dependencies that are required by this specific notebook are installed.

We first load this data set into a Panda's data frame. Then we peek at the data frame to make sure it is loaded correctly. Let's execute this code and view the results. We can see that the help desk data is correctly loaded. A document in natural processing is considered a collection of words or phrases that are related to a specific entity or topic.

For this example, we consider each question as a document. We need to convert each document into a document vector. A document vector is an array of all the words found in the document. We first need to clean up the questions before this conversion. For this, we first extract the question columns into a document variable

We create a function called process document that will cleanse and pre-process each document. In this function, we first convert all documents into lowercase. Then we remove stop words in the document using the remove stop words function in the NLTK package. We also remove the question mark character.

Then we split the document into individual words and return it. Now we call the process document function for each question or document in the documents variable. This returns a document vector which gets stored in doc_vectors. We print the second vector in the array and compare it to the original document.

Let's run this code now. We can see here how a question like, can Mac Air get infected by a virus, gets cleaned up and split into individual words.

In [2]:

```
from collections import defaultdict
from gensim import corpora
from gensim.parsing.preprocessing import remove_stopwords
import numpy as np
import os
import pandas as pd

#Read the input CSV into a Pandas dataframe
helpdesk_data = pd.read_csv("../input/datasets-in-it-ops-applied-ai/helpdesk_dataset.csv")

print("HelpDesk Data: ")
print(helpdesk_data.head())
```

```
HelpDesk Data:
                                        Question  \
0             My Mac does not boot, what can I do ?
1               Can Mac Air get infected by a Virus
2   My Mac is having boot problems, how do I fix it?
3                Do I need an anti virus on my Mac?
4  I have trouble connecting my monitor to my Mac...

                  LinkToAnswer
0  http://faq/mac-does-not-boot
1     http://faq/mac-book-virus
2  http://faq/mac-does-not-boot
3     http://faq/mac-book-virus
4  http://faq/mac-monitor-setup
```

In [3]:

```
#Extract the Question column
documents = helpdesk_data["Question"]

#Function to cleanse document
def process_document(document):

    #Remove stopwords, convert to lower case and remove "?" character
    cleaned_document = remove_stopwords(document.lower()).replace("?","")
    return cleaned_document.split()

#Create a document vector
doc_vectors=[process_document(document)
             for document in documents]

#Print the document and the corresponding document vector to compare
print(documents[1])
print(doc_vectors[1])
```

```
Can Mac Air get infected by a Virus
['mac', 'air', 'infected', 'virus']
```

3. Creating The LSI Model

Having prepared the data for LSI, we will now convert the document vector into an LSI model and measure similarities between the documents. This is a multi step process.

First, we need to create a dictionary based on the document vectors. The dictionary is a unique list of words found in these document vectors. To do this, we use the corpora.dictionary method. This generates a dictionary with words and corresponding identifiers. We will then print the dictionary. As we can see, each of the unique words in the document vector are listed with a corresponding identifying number.

Next, we need to convert the document vector into a corpus based on the identifiers in the dictionary. We use the doc2bow method to convert the vectors into this corpus. We print the document and the corpus for the second

document in the list to compare the results. As we can see, each word in the document is mapped to a tuple. The first number in the tuple is the word identifier in the dictionary. The second number is the total number of times this word appears in this document.

Now let's build the similarity index. To do this, we use the LSI model method found in the gensim package. We pass the corpus and the dictionary as parameters to the LSI model. Based on the model, we can then generate the similarity index by calling the matrix similarity method found in gensim.

We then print the similarity matrix. Let's run this code now. For each document in the input, the matrix lists the similarity code for this document with the other documents in the input. We have 10 input documents, so we get a 10 by 10 matrix.

For example, the second array lists the similarity score of the second document with all other documents in this corpus. Its similarity to itself is one. The higher the similarity, the more related these documents are. Now that we have built the similarity index, we can start predicting the question and recommending FAQs.

In [4]:
```
#Create the dictionary
dictionary = corpora.Dictionary(doc_vectors)

print("Dictionary created :")
dictionary.token2id
```

```
Dictionary created :
```

Out[4]:
```
{'boot,': 0,
 'mac': 1,
 'air': 2,
 'infected': 3,
 'virus': 4,
 'boot': 5,
 'fix': 6,
 'having': 7,
 'it': 8,
 'problems,': 9,
 'anti': 10,
 'need': 11,
 'connecting': 12,
 'help': 13,
 'mac.': 14,
 'monitor': 15,
 'trouble': 16,
 'boots,': 17,
 'error': 18,
 'shows': 19,
 'software': 20,
 'unsupporterd': 21,
 'connected': 22,
 'proper': 23,
 'resolution': 24,
 'flicker': 25,
 'monitor.': 26,
 'hdmi': 27,
 'use': 28,
 'connect': 29,
```

In [5]:
```
#Create a corpus
corpus = [dictionary.doc2bow(doc_vector)
          for doc_vector in doc_vectors]

#Review the corpus generated
print(doc_vectors[1])
print(corpus[1])
```

```
['mac', 'air', 'infected', 'virus']
[(1, 1), (2, 1), (3, 1), (4, 1)]
```

In [6]:
```
#Build the LSI Model
from gensim import models,similarities

#Create the model
lsi = models.LsiModel(corpus, id2word=dictionary)

#Create a similarity Index
index = similarities.MatrixSimilarity(lsi[corpus])

for similarities in index:
    print(similarities)
```

```
[ 9.9999994e-01  3.5355344e-01  2.8867516e-01  3.5355344e-01
  1.3759556e-08  2.8867513e-01 -3.0296217e-08  1.1460333e-08
  3.5355338e-01  4.0824828e-01]
[ 3.5355344e-01  1.0000000e+00  2.0412412e-01  5.0000000e-01
  1.6484243e-08  2.0412417e-01 -2.1429403e-10  2.8867510e-01
  2.5000000e-01  2.8867513e-01]
[ 2.8867516e-01  2.0412412e-01  1.0000000e+00  2.0412412e-01
 -2.0409514e-08  1.6666669e-01  3.0860671e-01  1.1203929e-08
  2.0412411e-01  2.3570226e-01]
[ 3.5355344e-01  5.0000000e-01  2.0412412e-01  9.9999994e-01
  1.6484243e-08  2.0412417e-01 -2.1429403e-10  2.8867513e-01
  2.5000000e-01  2.8867516e-01]
[ 1.3759556e-08  1.6484243e-08 -2.0409514e-08  1.6484243e-08
  1.0000000e+00  1.7943245e-08  3.3806172e-01  2.6950131e-09
  2.2360680e-01  7.9633944e-09]
[ 2.8867513e-01  2.0412417e-01  1.6666669e-01  2.0412417e-01
  1.7943245e-08  1.0000000e+00 -2.0282640e-08 -2.3956115e-09
  2.0412418e-01  2.3570226e-01]
[-3.0296217e-08 -2.1429403e-10  3.0860671e-01 -2.1429403e-10
  3.3806172e-01 -2.0282640e-08  1.0000000e+00 -7.1884396e-09
  1.8898220e-01 -2.1923015e-08]
[ 1.1460333e-08  2.8867510e-01  1.1203929e-08  2.8867513e-01
  2.6950131e-09 -2.3956115e-09 -7.1884396e-09  1.0000000e+00
  2.7179819e-08  2.7304045e-08]
[3.5355338e-01 2.5000000e-01 2.0412411e-01 2.5000000e-01 2.2360680e-01
 2.0412418e-01 1.8898220e-01 2.7179819e-08 9.9999994e-01 2.8867513e-01]
[ 4.0824828e-01  2.8867513e-01  2.3570226e-01  2.8867516e-01
  7.9633944e-09  2.3570226e-01 -2.1923015e-08  2.7304045e-08
  2.8867513e-01  1.0000000e+00]
```

4. Recommending FAQs

We will recommend FAQs based on the question being asked by a user. Let's say the user asks a question. I have boot problems with my Mac.

We first need to run this question through the same processing we did with the training dataset. We use the process document function to cleanse the question and then convert it into a corpus. We print the translated question.

Then we call the LSI method with this corpus as the index. It returns an equal and LSI model. Then we find the similarity of this model with all the other

questions in our training dataset. We do so by using the LSI model and calling it on the LSI index we built earlier. This returns the similarity scores for this question to all other documents in the training dataset.

We will print the similarity scores. As seen here, we get the similarity scores for this question with each of the documents. The scores are a tuple, with the first number indicating the document ID and the second number the similarity score. The higher the score, the more matching is this question to the document in the dataset. To find the top matching question, we do an argsort to sort the similarity scores based on the score and return the index of the document in descending order. We print this order. This gives the list of the most matching question to the least matching question. We can then print the similarity score and the corresponding question to check.

For recommending an FAQ, we pick the top question from this code list, find the corresponding FAQ link from the help desk data frame, and return the link. Let's run this code and review the results.

In [7]:
```
question = "I have boot problems in my Mac"

#Pre Process the Question
question_corpus = dictionary.doc2bow(process_document(question))
print("Question translated to :", question_corpus)

#Create an LSI Representation
vec_lsi = lsi[question_corpus]

#Find similarity of the question with existing documents
sims = index[vec_lsi]
print("Similarity scores :",list(enumerate(sims)))
```

```
Question translated to : [(1, 1), (5, 1)]
Similarity scores : [(0, 0.67856914), (1, 0.47982088), (2, 0.7835442), (3, 0.47982088), (4, 4.2491592e-09), (5, 0.39177212),
(6, -5.4133125e-08), (7, 0.0), (8, 0.47982085), (9, 0.55404943)]
```

Final Product Prototype

The Final Product Prototype will consist of the following components:

1. Frontend:
   The frontend will consist of a User interface where the user can type in the query or problem that he needs a solution or wants guidance with.
   The frontend will be simple and minimalistic.
   On submission of the query the backend model will analyze the query, find the best result and display the same to the user.
2. Backend
   The backend will consist of the ML model that is raking in the input from the user and then analyzing it to come up with the predicted solutions for the FAQs.

A brief snapshot of what the backend computing may look like is as follows:

```
Sorted Document index : [2 0 9 3 1 8 5 4 7 6]

 ------------------------------------------------------------

0.7835442  ->  My Mac is having boot problems, how do I fix it?
0.67856914  ->  My Mac does not boot, what can I do ?
0.55404943  ->  Can I connect two monitors to my Mac?
0.47982088  ->  Do I need an anti virus on my Mac?
0.47982088  ->  Can Mac Air get infected by a Virus
0.47982085  ->  Can I use a HDMI monitor with my Mac?
0.39177212  ->  When my Mac boots, it shows an unsupporterd software error
4.2491592e-09  ->  I have trouble connecting my monitor to my Mac. Can you please help?
0.0  ->  I see a flicker in my monitor. Is that a virus?
-5.4133125e-08  ->  My Monitor does not show in proper resolution when connected to my Mac. How do I fix it?

 ------------------------------------------------------------

Recommended FAQ : http://faq/mac-does-not-boot
```

Product Details

The user will input the query, the system will accept the query, analyze it and then return the corresponding link which shall solve the problem

This will save time browsing through FAQs and also the time saved in ringing up the helpline system.

Code Implementation

The code has been implemented using Python Language in Jupyter Notebook IDE for visual reference.

The complete code along with the dataset is available at the following GitHub Links:

Main Link: https://github.com/aryashah2k/Feynn-Labs

Assignment Specific Link:

Name	Link
Dataset	https://github.com/aryashah2k/Feynn-Labs/blob/main/helpdesk_dataset.csv
Notebook	https://github.com/aryashah2k/Feynn-Labs/blob/main/faq-recommendation-for-helpdesk.ipynb

Conclusion

As seen from the results, the top document match is document number two, followed by document number zero.

Comparing the questions, we can see that the top two matches were questions regarding boot problems, which the user question is about. This shows that the algorithm works well. We also see that the right FAQ has been pulled up in the link shown. This FAQ can then be passed to the user.

Please note that this is a small dataset for demonstration purposes. Real world examples use much larger datasets. We use the question as the document for training, but we can instead use the entire content of the FAQ article as the document also. This would require a lot more processing, but can lead to more accurate results.

References

1. https://www.kaggle.com/datasets/aryashah2k/datasets-in-it-ops-applied-ai
2. https://thebotplatform.com/solutions/hr-people/helpdesk-and-faqs/
3. https://en.wikipedia.org/wiki/Latent_semantic_analysis
4. https://medium.com/analytics-vidhya/nlp-with-latent-semantic-analysis-b3de6e16ad7d
5. https://towardsdatascience.com/latent-semantic-analysis-intuition-math-implementation-a194aff870f8
6. https://medium.com/acing-ai/what-is-latent-semantic-analysis-lsa-4d3e2d18417a
7. https://radimrehurek.com/gensim/
8. https://www.analyticsvidhya.com/blog/2021/05/topic-modelling-in-natural-language-processing/