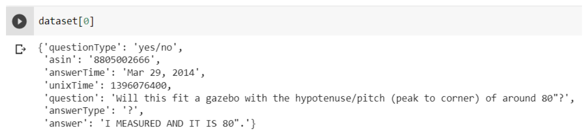
Dataset Introduction

We have chosen the **Amazon Question and Answer Data** to perform predictive tasks. In particular, we selected the dataset **Patio Lawn and Garden** with 59,595 questions.
We have decided to pick the Amazon Question and Answer Data because we are interested in seeing the best way to categorize data. Looking at the data, we can see there are 7 main categories:

- ‘questionType’: either yes/no or open-ended
- ‘asin’: uniquely identifies a product,
- ‘answerTime’: represents the time the answer was given
- ‘unixTime’: represents the time the answer was given.
- ‘question’: contains a string that represents the question
- ‘answerType’: contains 3 possible options: ”?”(unknown), “yes/no”, “free-response”
- ‘answer’: contains a string that represents the answer to the question.

Below is an example of a data point:



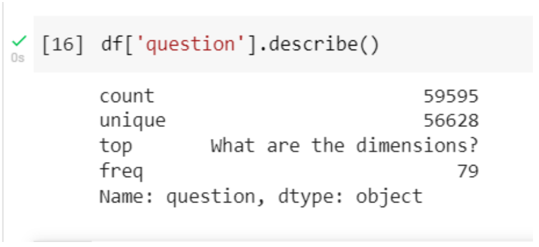
```
dataset[0]
```

```
{'questionType': 'yes/no',
 'asin': '8805002666',
 'answerTime': 'Mar 29, 2014',
 'unixTime': 1396076400,
 'question': 'Will this fit a gazebo with the hypotenuse/pitch (peak to corner) of around 80"?',
 'answerType': '?',
 'answer': 'I MEASURED AND IT IS 80".'}
```

Exploratory Analysis

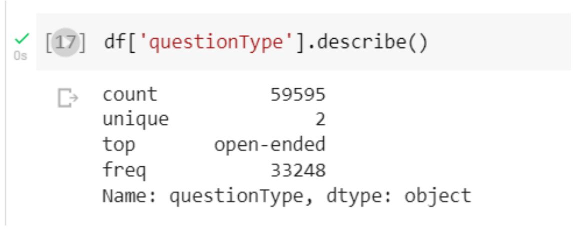
Data Distribution

In this dataset, there are 59,595 questions with 56,628 of them being unique. It is apparent that most of the questions are unique, so we have a good amount of data to work with only a little filtering. We can also see that roughly half of the questions are yes/no and the other half are open-ended.



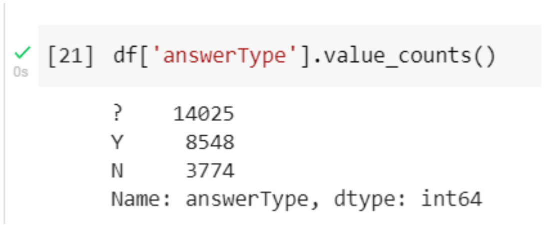
```
[16] df['question'].describe()
```

```
count                         59595
unique                        56628
top       What are the dimensions?
freq                             79
Name: question, dtype: object
```



```
[17] df['questionType'].describe()
```

```
count           59595
unique              2
top        open-ended
freq            33248
Name: questionType, dtype: object
```

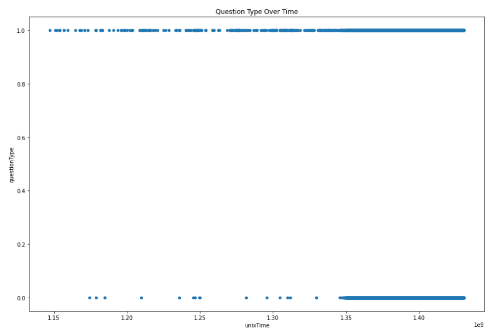


```
[21] df['answerType'].value_counts()
```

```
?    14025
Y     8548
N     3774
Name: answerType, dtype: int64
```

Observation 1

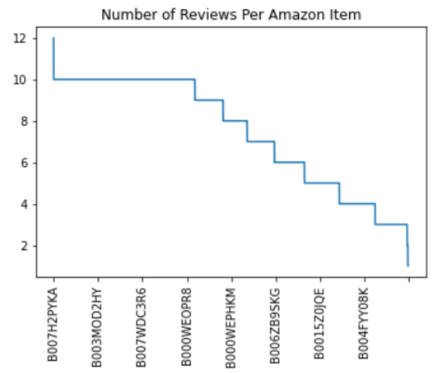
Next, let’s look at the Question Type over time. In the diagram below, 1 represents an open question, and 0 represents a yes/no question.



As we can see, there are more questions in the past two years than there have been in the time period between 2 and 7 years ago. In addition, a majority of the yes/no questions started within the past two years while the open-ended questions have been steadily increasing but it has been consistently common at all times.

Observation 2

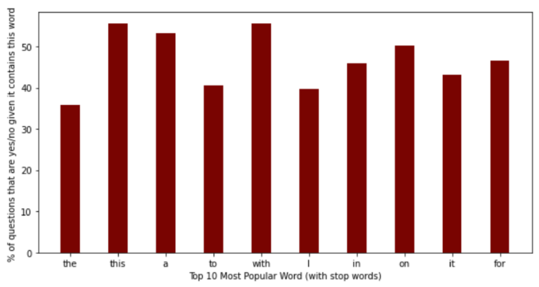
Here, we can see that the maximum number of reviews for the same Amazon Item is 12 and the minimum is one.



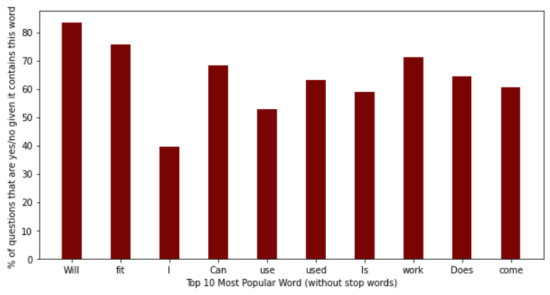
However, it is not hard to see that there exist very few instances of having multiple questions per item as only eight items contain more than two questions. In addition, the majority of these questions for the same item are open-ended questions. This may be explained by the category that this dataset is in (Patio Lawn and Garden). It is possible that yes/no questions will be more prominent in the technology category as its products contain more diverse features and the buyers are more incentivized to ask yes/no questions to confirm if a feature is present.

Observation 3

We have learned that popular words can be a great measure for various predictive tasks from past assignments and lectures. Let's look at how predictive the top 10 most popular words are. The diagrams below show the percentage of time in which a popular word is used in a question and its question type is yes/no. To determine the effects of the stopwords, we created one diagram which includes stopwords and one diagram which doesn't include the stopwords.



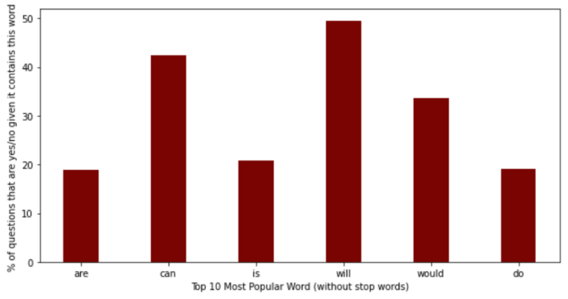
In the diagram above, we included stopwords, and it is apparent that stopwords dominated the top 10 most popular words. Roughly 50% of the time or lower, the questions that include these words are yes/no questions. It is apparent that stopwords are likely not a very useful factor to take into account when performing predictive tasks.



The diagram above excludes stopwords when calculating the top 10 most popular words. We can see that the accuracy is fairly higher compared to stopwords which also have high popularity. Most of the words appear in a yes/no question roughly 60% of the time or higher. From this observation, it is quite clear the removal of stopwords will likely increase the accuracy of our predictive model. If we examine the words from the second diagram more deeply, we can notice that some words are not as predictive as others. An example can be seen in the word “I” which appears in a yes/no question roughly 40% of the time. We can expand our exploratory analysis based on the information collected here by hand-selecting words that we think are relevant from or based on these words and performing a more in-depth analysis of them.

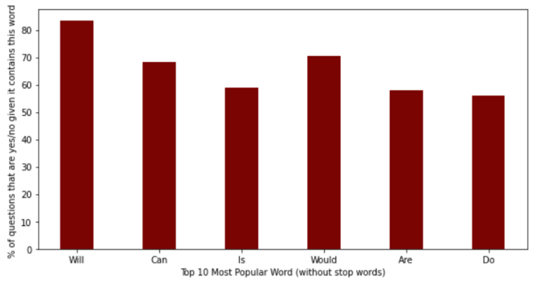
Observation 4

Based on the information we have gathered from the top 10 words without stopwords, we have hand-selected 5 words that we think will be more effective to predict the question type. Yes/no questions commonly start with a set of specific words such as “does”, “do”, “will”, and more. We performed the same analysis from Observation 3 to determine how effective these words are for predicting the question type.



The diagram above shows the percentage of time in which these 5 words are used and the question type is yes/no. Based on the analysis, we noticed that these words are surprisingly unlikely to appear in yes/no questions.

Instead of ignoring the capitalization of the words shown in the first diagram, we decided to only examine these words only if their first letter is capitalized. The diagram below shows the result of the improved analysis.



As we can see from the second diagram, the percentage of time in which these words appear in yes/no questions significantly increased after taking capitalization into account.

Predictive Task

Our predictive task is to predict question types based on the question. For this predictive task, we have considered some models that we learned from class. We have attempted on using the top 10 most popular words, Naive Bayes, and TF-IDF. For using the top 10 most popular words, we have explored different options to increase the projected accuracy. Some of the features that may become relevant are the ‘answerTime’ and ‘question’. For questions, we can implement the models mentioned above, and the ‘answerTime’ can become useful for temporal dynamics for our recommendation model.

In order to assess the validity, we split our dataset into 2 sets: training and testing set. The training set consists of 75% of the dataset, and the testing set consists of 25% of the dataset. We verify the validity by training our models using the training set and predicting using the testing set. We then compute the accuracy by comparing the result of our prediction with the actual data from the testing set.

For our baseline, we used Logistic regression by counting the total number of keywords that were present in each question. These keywords were hand-selected words by us that we thought would indicate a yes/no type of question. These keywords are “would”, “will”, “can”, “is”, “are”, “do”.Through our exploratory analysis, we saw that some words within the top 10 popular words are more effective than others which we can use to improve our baseline model

Using the process we did in the exploratory analysis, we found the top 20 most popular words and hand-selected 5 words that we think are the most effective for this predictive task. In particular, to find the most popular words, we created a dictionary that stores the frequencies of each word. By sorting the frequencies, we were able to identify the most popular words.

For the hand selection among the 20 words, we focused on finding the common starting word for yes/no questions such as “Does”, “Will”, etc. To verify our speculation, we also computed the percentage of these words appearing in yes/no questions using our training set as we did in the exploratory analysis.

We will compare the accuracies between this model and the baseline model to determine if an improvement was achieved.

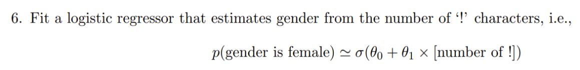
On the other hand, we also plan to use the TF-IDF model. For this model, we will remove all punctuation while keeping capitalization as we believe that capitalization can play a major factor in our model. Then we perform lemmatization on these words for concentrated term frequency. Lastly, we transform these strings into a document-term matrix for the model to train.

We will also compare the accuracy between the baseline model and the TF-IDF model to determine if an improvement was made.

Models

1. Baseline (Hand-Selected Words)

This baseline uses hand-selected words that we believed to indicate yes/no questions without any prior analysis of the dataset. In particular, we used the count of “would”, “will”, “can”, “is”, “are”, “do” as the six features in our feature vector to make predictions. However, these words that we arbitrarily selected may still be very ineffective. In fact, these words did turn out to be relatively less effective and we later made some further improvements on top of this model. (see number 3)


6. Fit a logistic regressor that estimates gender from the number of ‘!’ characters, i.e.,

$$p(\text{gender is female}) \simeq \sigma(\theta_0 + \theta_1 \times [\text{number of !}])$$

In homework 1, we have seen that models similar to this can be quite effective for classification problems (see screenshot above). Rather than predicting the gender classification using the number of exclamation marks, we instead used the number of selected words as our feature vector. Similar to homework 1, we are performing a binary classification (yes/no vs open-ended), and we expect this model to serve as a good starting/baseline model. We also learned from later homework that a small increase in dimension can increase the effectiveness of our models, and that is the reason behind the fact that we decided to select 6 words instead of just 1.

```
feat = [1,
datum['question'].count("would"),
datum['question'].count("will"),
datum['question'].count("can"),
datum['question'].count("is"),
datum['question'].count("are"),
datum['question'].count("do")]
```

2. K-Nearest Neighbor

Through some research on classification problems, we learned that the K-Nearest Neighbor model can be effective for our predictive task. This model loads the training data points and calculates the “distance” between them. Based on these “distances,” the model simulates the idea of similarity between the data points.

Due to its nature of determining similarities, we expected the K-Nearest Neighbor model to be a great tool for classification predictions. Despite its high cost, this lazy learning model should be effective for predicting relatively simple patterns. Evidently, it yields better predictions compared to the baseline model after some adjustments to the K value.

There does exist a scalability problem with this model. It becomes significantly slower as the dataset size grows compared to all the other models that we are. However, we still chose to use this model because we recognized the fact that the size of our dataset won’t lead to any problem with scalability.

3. Most Popular Words (Words Selected Based on Exploratory Analysis)

This model is an improvement on top of the baseline model. Instead of arbitrarily selecting words that we think are representative of the question type, we utilized information that we collected from our exploratory analysis. In particular, we found the top 20 most popular words and examined the percentage of them appearing in a yes/no question vs open-ended questions.

By referencing these data and some common grammatical patterns, we came up with 5 different words for our feature vector, which led to a huge improvement compared to the baseline model. Through various experimentations with our feature vectors, we came to the conclusion to decrease the feature dimension by 1 for improved accuracy.

We worried that this model may have some overfitting problem, and that is the reason that we didn’t purely select the top 5 most popular words. Instead, we took common grammatical patterns and capitalization into account that extends outside of our training in hope of avoiding overfitting.

```
feat = [1,
datum['question'].count("Does"),
datum['question'].count("Is"),
datum['question'].count("Will"),
datum['question'].count("Can"),
datum['question'].count("Do")]
```

4. Naive Bayes

This is a conditional probability model based on Bayes’ Theorem. It was briefly mentioned during the lecture and some of the past slides.

$$p(C_k \mid \mathbf{x}) = \frac{p(C_k)\, p(\mathbf{x} \mid C_k)}{p(\mathbf{x})},$$

where C_k represents the possible outcomes and x represents the feature vector.

Naive Bayes is commonly used for in-text classification and is known for its high performance and accuracy in multi-class classification problems. For this exact reason, we chose this model as our predictive task is a simpler binary classification problem and therefore we expect a similarly high rate of success and performance from Naive Bayes. And the predictions that this model produces are still more accurate than all the previous models that we have tested.

5. TF-IDF

TF-IDF is a model that we have explored for Assignment 1. It is especially suitable for this predictive task due to the text-based form factor (question text) that our input takes.

This model takes account of term frequency and offsets relatively less useful terms (such as “of”, “as”, etc.) when computing predictions. (using inverse document frequency)

How the inverse document frequency minimizes the weight of irrelevant frequent terms is that it finds frequent terms that also appear in an English corpus and takes its inverse.

TF-IDF trains itself using every term within the input question text and its frequencies instead of just a limited number of terms that we implemented for our baseline model. Having more information to train the model, we believe that TF-IDF will perform significantly better than our base model. And from what we have seen, it yields highly accurate predictions.

6. Unsuccessful Attempts - Number of Question Marks

We have experimented with other models, but they have produced rather unsuccessful results. In particular, we have tried a modified classification model from homework 1 which uses the number of question marks (instead of exclamation marks like on the homework) as the feature vector. However, it yields a very low accuracy as question marks are not necessarily a good indicator of question types.

Literature

This dataset came from Amazon and was collected by Menting Wan and Julian McAuley by crawling Amazon. In this study, they were attempting to test a series of methods to model the ambiguity and subjectivity in product-related opinion question-answering systems. Another study done by Wan and McAuley on the same dataset was creating a system called Moqa, which is a classifier of whether a response is relevant to a question. Similar datasets have been used for NLP-text-classification in order to understand the tone of the text.

One classification technique used by both of McAuley’s research papers is Mixture of Experts (MoE), which combines several weak classifiers in order to generate predictions. There are multiple types of MOE, including s-MOE, which only includes the top-voted answers for training and is considered state of the art, m-MOE, which includes all answers for each question, and finally, m-MOE-S which includes subjective information in an attempt to help the classification.
The conclusions from other studies are not relevant to our work because they are either looking for the best response to a question or dividing their questions based on this. In contrast, we are looking for a way to tell between yes/no questions and open-ended questions.

Result & Conclusion

Models	Accuracy
Number of Question Marks	0.5714765100671141
Baseline (Hand-Selected Words)	0.6228187919463087
K-Nearest Neighbor	0.6860738255033557
Most Popular Words (Words Selected Based on Exploratory Analysis)	0.6959731543624161
Naive Bayes	0.7686241610738255
TF-IDF	0.8865771812080537

It is quite apparent that TF-IDF worked the best for this particular dataset. Unlike our baseline model and the most popular words model, TF-IDF takes more words into account when training its model. For our baseline model and the most popular words model, we only take into account 5 specific words that we believe to be relevant for this predictive task. However, the TF-IDF model records all term frequencies while minimizing the weight of less useful words.

Usually, stopwords can hinder the result of the TF-IDF model, but we noticed that removing stopwords in fact decreases the accuracy of the prediction for this dataset. Because of that, we decided to keep the stopwords when training our model.

On the other hand, the Naive Bayes model is limited by its assumption of independence, but we expected better results given the fact that we are dealing with in-text classification. In the end, we realize that the Naive Bayes model assigns zero probability to categorical variables that are not available in the training set, which can lead to a drop in accuracy.

The K-Nearest Neighbor model performed similarly to the most popular words model. Though both of them performed worse than Naive Bayes and TF-IDF. It is not as surprising since the way the K-Nearest Neighbor model computes the similarity (distance) between data points is not as effective as the conditional probability model, not to mention TF-IDF which takes account of term frequency and inverse document frequency.

Lastly, the number of question marks model has very apparent flaws as it is not hard to see that the number of question marks is not strongly related to the corresponding question type.

For the number of question marks model, baseline model, and most popular words model, we have used a similar feature vector. On the other hand, Naive Bayes, K-Nearest Neighbor, and TF-IDF models used a different type of feature vector that contains far more information than the other feature vector. It is not surprising that the latter three models produce better predictions.