COMP3074 Human-AI Interaction
Lab 5: The Art of Evaluation

Jeremie Clos

December 2023

Contents

1 Introduction

This week's lab is focused on evaluation. Evaluation is what allows you to know what needs to be improved for your conversational AI to perform better when deployed in the field. For the purpose of this lab, we will separate our evaluation in three components:

- **Performance** (or system evaluation), focusing on testing isolated components, typically through the use of benchmarking exercises;
- **Usability** (or user evaluation), focusing on testing the experience of the user when using the system, typically by using actual participants;
- **Ethics** (or RRI evaluation), focusing on evaluating the potential impact of the system on its wider environment, through the use of thought experiments.

Note that the section on statistical testing is simplified for the sake of the lab, do not come at me if you are a data science student!

2 Performance testing

Performance testing focuses on the evaluation of objective but not necessarily representative metrics of the system. This usually requires isolating components of the system and evaluating them individually. The standard tasks that elements of a chatbot perform are the following:

- **Routing**: given a user input, route them towards the correct outcome. For example, detect whether a user accepts or rejects (yes/no) a statement. This is essentially a **classification** task and is usually measured using classification Accuracy, F_1-Score, and Precision/Recall[1].
- **Search**: given a user input, search for the most similar item in a corpus. This is a **retrieval** task, and as such measures such as Precision, Recall, and NDCG can be used to test the performance of that subsystem.
- **Entity recognition**: given a user input, find names of entities of interest. This is a **tagging** task and is usually evaluated quite similarly to a classification task, using Accuracy, F_1-Score, and Precision/Recall. You can consider entity recognition in the context of chatbots as a restricted version of the evaluation of Named Entity Recognition[2].

2.1 Choosing metrics

Choosing metrics is the first step in evaluating a system. While it is tempting to think that you can just compute everything and call it a day, you need to pay attention to whether the metrics you are choosing are actually aligned with what you are measuring. For example, when evaluating the intent classification component of a system, does recall really matter? Or is the accuracy more representative of a well-performing intent detection system?

2.2 Experimental design for classification

Experimental design is how you will compute the chosen measure of performance. As seen in the lectures, the train/test split and cross-validation are the two preferred methods for generalisation estimation for classification-based systems, while bootstrapping is much less popular but can also lead to interesting insights.

[1] `https://learn.microsoft.com/en-us/azure/ai-services/language-service/custom-text-classification/concepts/evaluation-metrics`

[2] `https://learn.microsoft.com/en-us/azure/ai-services/language-service/custom-named-entity-recognition/concepts/evaluation-metrics`

2.2.1 Metrics for classification

- **Accuracy**: Accuracy is a simple and easy to understand metric that measures how many instances are correctly predicted in comparison to the total number of instances. It is most useful when the classes being evaluated are evenly distributed and the consequences of misclassification are the same.
- **Precision** and **Recall**: The concept of precision is especially relevant in scenarios where the consequences of incorrect positive predictions are considerable. On the other hand, recall, also known as sensitivity, is the ratio of true positive predictions to the actual number of positive cases. This is especially important in cases where the effects of false negatives are significant, requiring a high rate of positive cases to be identified.
- **F1-Score**: The F1-score is a useful tool when a balance between precision and recall is needed. It is especially beneficial in situations where it is important to avoid false positives while still capturing true positives. The F1 score is the harmonic mean of precision and recall, making it an ideal metric for these types of scenarios.
- **ROC-AUC**: The Receiver Operating Characteristic (ROC) curve, along with the Area Under the Curve (AUC), are used together to evaluate a classifier's ability to differentiate between classes. An elevated AUC value is indicative of a model's proficiency in correctly classifying 0s as 0s and 1s as 1s, thereby reflecting its discriminative power.
- **Confusion Matrix**: This matrix gives an overview of true positives, false positives, true negatives, and false negatives. It offers a precise illustration of the classifier's performance, allowing you to see where it is making mistakes and detect potential issues in the training process.

2.2.2 The train/test split

For demonstration purposes, let us assume we are working with a simple dataset for sentiment analysis, and we will use a Naive Bayes classifier. The following code provides an example of how to do this:

```
import numpy as np
from sklearn.datasets import fetch_20newsgroups
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report, confusion_matrix,
    accuracy_score

# Load dataset
data = fetch_20newsgroups(subset='all', categories=['sci.space', 'comp.graphics'
    ])
X, y = data.data, data.target

# Split dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
    random_state=42)

# Text preprocessing and vectorisation
vectorizer = TfidfVectorizer(stop_words='english')
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)

# Train the classifier
classifier = MultinomialNB()
classifier.fit(X_train_vec, y_train)

```

```
# Predict on the test set
y_pred = classifier.predict(X_test_vec)

# Evaluate the classifier
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

2.2.3 Cross-validation

To compare multiple classifiers using 10-fold cross-validation, we can extend the previous code to iterate over different classifier models. For this example, I will include three different classifiers: Multinomial Naive Bayes, Support Vector Machine (SVM), and Random Forest. Here's how you can modify the code:

```
import numpy as np
from sklearn.datasets import fetch_20newsgroups
from sklearn.model_selection import cross_val_score, StratifiedKFold
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import make_pipeline

# Load dataset
data = fetch_20newsgroups(subset='all', categories=['sci.space', 'comp.graphics'
    ])
X, y = data.data, data.target

# Define classifiers to evaluate
classifiers = {
    "Multinomial Naive Bayes": MultinomialNB(),
    "Support Vector Machine": SVC(),
    "Random Forest": RandomForestClassifier()
}

# Define the k-fold cross-validator (k=10)
kfold = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

# Iterate over classifiers
for name, classifier in classifiers.items():
    pipeline = make_pipeline(TfidfVectorizer(stop_words='english'), classifier)
    scores = cross_val_score(pipeline, X, y, cv=kfold, scoring='accuracy')
    print(f"{name}:")
    print("  Cross-validation scores:", scores)
    print("  Average accuracy:", np.mean(scores))
    print()
```

2.2.4 Bootstrapping

To compare multiple classifiers using bootstrapping, we need to modify the evaluation strategy. Bootstrapping[3] involves repeatedly sampling from the dataset with replacement to create multiple "bootstrap" datasets. Each classifier is then trained and evaluated on these datasets.

```
import numpy as np
from sklearn.utils import resample
from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import make_pipeline
```

[3] https://en.wikipedia.org/wiki/Bootstrapping_(statistics)

```
from sklearn.metrics import accuracy_score

# Load dataset
data = fetch_20newsgroups(subset='all', categories=['sci.space', 'comp.graphics'])
X, y = data.data, data.target

# Define classifiers to evaluate
classifiers = {
    "Multinomial Naive Bayes": MultinomialNB(),
    "Support Vector Machine": SVC(),
    "Random Forest": RandomForestClassifier()
}

# Number of bootstrap samples - the higher it is, the more experiments will be run and the longer it will take
n_iterations = 10
size = int(len(X) * 0.50)  # 50% of the dataset

# Iterate over classifiers
for name, classifier in classifiers.items():
    scores = []
    for i in range(n_iterations):
        # Prepare bootstrap sample
        X_sample, y_sample = resample(X, y, n_samples=size)

        # Vectorize text data
        vectorizer = TfidfVectorizer(stop_words='english')
        X_sample_vec = vectorizer.fit_transform(X_sample)

        # Train classifier
        classifier.fit(X_sample_vec, y_sample)

        # Evaluate classifier
        X_test_vec = vectorizer.transform(X)  # Transform the entire dataset for testing
        y_pred = classifier.predict(X_test_vec)
        score = accuracy_score(y, y_pred)
        scores.append(score)

    # Display results
    print(f"{name}:")
    print("  Average accuracy:", np.mean(scores))
    print()
```

2.3 Experimental design for question answering systems

Evaluating a question-answering system in Python can be approached by focusing on accuracy, the simplest and most direct metric. The process involves comparing the system's answers to a set of predefined questions against the correct answers. Here is a basic outline of how to do it:

1. **Prepare a test set**: you need a set of questions and the corresponding correct answers. This can be a small set if you are looking for simplicity. You can group yourself up with a list of classmates or friends in order to come up with a set of questions/answers if you are using the same dataset. Make sure that you are not simply copying from your actual dataset, or else the evaluation would have no point.

2. **Get system answers**: run these questions through your system to get the answers.

3. **Compare answers**: for each question, compare the system's answer with the correct answer. You might want to consider whether the comparison should be exact (string

equality) or whether you need something more flexible, such as comparing the cosine similarity between the mean average word embedding of your answer and of the correct answer. If you are doing a more flexible evaluation, you will need to establish a threshold of similarity that is considered sufficiently high to trigger a "correct" classification.

4. **Calculate accuracy**: Accuracy is the number of correct answers divided by the total number of questions. Other potential metrics you could use are Precision and Recall[4] and NDCG[5].

```python
# Example test set of questions and correct answers
test_set = [
    {"question": "What is the capital of France?", "correct_answer": "Paris"},
    {"question": "Who wrote Macbeth?", "correct_answer": "William Shakespeare"},
    # Add more questions and answers here
]

# Function to get answer from your QA system
def get_answer(question):
    # This function should be replaced with the actual method to get an answer from your QA system
    return "Dummy Answer"  # Placeholder

# Evaluate the QA system
correct_count = 0
for item in test_set:
    system_answer = get_answer(item["question"])
    if system_answer.strip().lower() == item["correct_answer"].lower():
        correct_count += 1

# Calculate and print accuracy
accuracy = correct_count / len(test_set)
print(f"Accuracy: {accuracy:.2f}")
```

2.4 Running statistical tests

Statistical testing is a procedure that we use to determine whether an effect observed in data is statistically significant or likely to be due to random chance. There are two categories of statistical tests: parametric tests (which require the knowledge that the data is distributed in a certain way) and nonparametric tests (which do not have as many assumptions). Because we do not have a lot of information about the distribution of performance scores, let us assume that we should use a nonparametric test. The testing procedure is divided in two steps: (1) a diagnostic test is run over the results of N systems to see whether they all perform the same (here the Kruskal-Wallis test) ; (2) if not, a post hoc test is run to see which performs better/worse (here the Dunn confirmation test).

Before moving on, make sure to get familiar with the notion of a p-value. A good definition can be found in the literature [1], copied below:

> The P value is defined as the probability under the assumption of no effect or no difference (null hypothesis), of obtaining a result equal to or more extreme than what was actually observed. The P stands for probability and measures how likely it is that any observed difference between groups is due to chance. Being a probability, P can take any value between 0 and 1. Values close to 0 indicate that the observed difference is unlikely to be due to chance, whereas a P value close to 1 suggests no difference between the groups other than due to chance. Thus, it is common in

[4] https://en.wikipedia.org/wiki/Precision_and_recall

[5] https://towardsdatascience.com/demystifying-ndcg-bee3be58cfe0

medical journals to see adjectives such as highly significant or very significant after quoting the P value depending on how close to zero the value is.

In the context of comparing systems, it means that a low p-value can be interpreted as a high likelihood that the difference in system performance is genuine, while a high p-value can be interpreted as a high likelihood that the difference in system performance is due to random chance. Let us take as an example a comparison of the accuracies of 3 classifiers in a 10-fold cross-validation experiment. For a nonparametric approach to compare the results of three classifiers from a cross-validation study, you can use the Kruskal-Wallis test, which is the non-parametric alternative to ANOVA. This test is appropriate when the data is not known to meet the assumptions of normality and homogeneity of variances that would be necessary for a t-test or an ANOVA.

After the Kruskal-Wallis test, if a significant difference is found (we fix the threshold at 0.05), you can perform post hoc pairwise comparisons using a nonparametric post hoc test like the Dunn's test. However, note that Python's standard libraries do not include Dunn's test, so you might need to use additional packages like scikit-posthocs.

```
pip install scikit-posthocs
```

```
import numpy as np
from scipy.stats import kruskal
import scikit_posthocs as sp

# Example accuracies from 10-fold cross-validation for 3 classifiers
acc_classifier_1 = np.array([0.7, 0.7, 0.7, 0.9, 0.9, 0.9, 0.8, 0.9, 0.9, 0.8])
acc_classifier_2 = np.array([0.4, 0.4, 0.4, 0.4, 0.5, 0.2, 0.3, 0.2, 0.5, 0.4])
acc_classifier_3 = np.array([0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1])

# Kruskal-Wallis test
stat, p = kruskal(acc_classifier_1, acc_classifier_2, acc_classifier_3)
print("Kruskal-Wallis test results: H =", stat, ", p-value =", p)

# Post-hoc analysis if Kruskal-Wallis test is significant
if p < 0.05:
    # Combine the groups for post-hoc analysis
    all_accuracies = np.concatenate([acc_classifier_1, acc_classifier_2,
    acc_classifier_3])
    groups = ['classifier_1'] * len(acc_classifier_1) + ['classifier_2'] * len(
    acc_classifier_2) + ['classifier_3'] * len(acc_classifier_3)

    # Post-hoc pairwise comparisons using Dunn's test
    dunn_test_results = sp.posthoc_dunn([acc_classifier_1, acc_classifier_2,
    acc_classifier_3], p_adjust='holm')
    print("Dunn's test post-hoc pairwise comparisons:\n", dunn_test_results)
```

Running this with the provided values should yield the following result:

```
Kruskal-Wallis test results: H = 27.083819752509932 , p-value = 1.314689717756897
    e-06

Dunn's test post-hoc pairwise comparisons:
              1         2           3
1  1.000000e+00  0.018531  5.844653e-07
2  1.853065e-02  1.000000  1.853065e-02
3  5.844653e-07  0.018531  1.000000e+00
```

The first line tells us that the classifiers are not all performing the same, which leads us to run the post-hoc test. that the You can see that we have a very low p-value between each system pair, meaning that they are all probably performing very differently (which we knew because I just made up the numbers - you will need to run tests with your own data). One thing to note, however, is that we had to run the test over 10 cross-validation results - this is because we need several paired observations to gather the evidence of a difference being genuine. The

more paired observations we have, the smaller the difference in performance needed to achieve statistical significance. For example, if we had run a 1,000-fold cross-validation, a 10% (0.1) difference could have been significant, while on a 10-fold cross-validation a larger one is needed.

2.5 Reporting your results

A nice way to report your results when conducting complex experiments is to use graphical representations, such as box plots[6], to visualise the differences between the systems you are comparing. Going back to our comparison of three classifiers, let us use the Seaborn and Matplotlib libraries to display some nice visualisation of our classifiers' performance.

```
pip install seaborn, matplotlib, pandas, numpy
```

```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

# Artificial data for three classifiers
np.random.seed(10)  # For reproducibility
data = {
    "Classifier A": np.array([0.75, 0.75, 0.74, 0.92, 0.69, 0.39, 0.85, 0.69,
    0.79, 0.68]) ,  # 10-fold CV scores
    "Classifier B": np.array([0.34, 0.43, 0.34, 0.24, 0.15, 0.32, 0.43, 0.22,
    0.45, 0.44]),
    "Classifier C": np.array([0.13, 0.53, 0.14, 0.13, 0.15, 0.16, 0.18, 0.31,
    0.21, 0.41])
}

# Convert data to a Pandas DataFrame
df = pd.DataFrame(data)

# Long format for Seaborn
df_long = pd.melt(df, var_name='Classifier', value_name='Accuracy')

# Create the boxplot
sns.boxplot(x='Classifier', y='Accuracy', data=df_long)

# Add title and labels
plt.title('Cross-Validation Accuracy of Three Classifiers')
plt.ylabel('Accuracy')
plt.xlabel('Classifier')

# Show the plot
plt.show()
```

When running this code, you should get a figure similar to the one shown in Figure 1.

2.6 Activity

Identify the components of your chatbot and design an evaluation plan. What metrics will you collect? Why? What do they represent for the end-user? How will you measure them?

3 Usability testing

Usability testing is another facet of the evaluation of a conversational AI, focusing on the user experience.

[6] https://www.ncl.ac.uk/webtemplate/ask-assets/external/maths-resources/statistics/data-presentation/box-and-whisker-plots.html

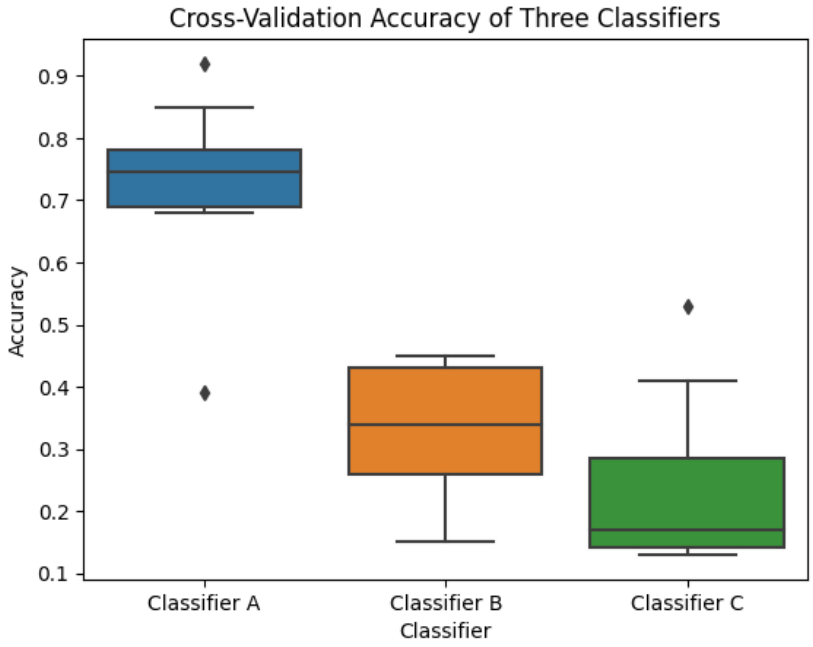

Figure 1: Comparison of classifiers A, B, and C

3.1 Chatbot Usability Questionnaire

Usability questionnaires, such as the Chatbot Usability Questionnaire[7] are a common way to assess the usability of software applications. The person driving the experiment typically hands them to the participants after a set period of time using the system to perform a set of predefined tasks. The forms, using a list of criteria on a scale of 1 to 5 (Likert scale), help determine the users' opinion of the chatbot's usability. Here is a potential structure for a very simple chatbot usability experiment:

1. Introduction: the experimenter introduces their system to the participants, explaining its purpose and functionalities.
2. Task assignment: participants are handed a list of tasks that they must accomplish using the chatbot and are given the opportunity to ask clarification questions.
3. Intervention: participants are given a set amount of time (e.g., 20 minutes) to perform the tasks. They do not have to finish all the tasks, so there is no element forcing them to work fast.
4. Questionnaire hand-in: the experimenter hands participants the questionnaire and asks them to fill it in.

3.2 Finding participants

Much like all evaluations, usability testing requires you to find participants which fit the profile of the actual users you have in mind. For example, you would not test a chatbot meant to help elderly dementia patients using a group of university students. This is the concept of *ecological*

[7] `https://www.ulster.ac.uk/research/topic/computer-science/artificial-intelligence/projects/cuq`

validity[8]: a study's variables and conclusions must be relevant to its population and context. Of course, that does not mean that you need to match this population perfectly. If you have no reason to believe that there is a meaningful difference between your test population (the participants) and your target (the users), then you can use this population for your usability experiment.

3.3 Reporting your results

In addition to summary statistics such as the CUQ scoring system[9] you can also use data visualisation tools to synthesise the results of your evaluation. Two common examples are a heatmap and a stacked bar chart.

3.3.1 Visualisation through heatmap

Heat maps[10] are a popular way to visualise information. They use a gradient of colour to represent the magnitude of a result, where a darker shade usually represents a higher value.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Sample data with responses from three participants for 16 questions
data = {
    'Question 01': [1, 2, 3],
    'Question 02': [2, 3, 4],
    'Question 03': [3, 4, 5],
    'Question 04': [4, 5, 1],
    'Question 05': [5, 1, 2],
    'Question 06': [1, 2, 3],
    'Question 07': [2, 3, 4],
    'Question 08': [3, 4, 5],
    'Question 09': [4, 5, 1],
    'Question 10': [5, 1, 2],
    'Question 11': [1, 2, 3],
    'Question 12': [2, 3, 4],
    'Question 13': [3, 4, 5],
    'Question 14': [4, 5, 1],
    'Question 15': [5, 1, 2],
    'Question 16': [1, 2, 3]
}

df = pd.DataFrame(data)

# Convert DataFrame to long format for easier plotting
df_long = df.melt(var_name='Question', value_name='Response')

# Create a pivot table for the heatmap
response_pivot = df_long.pivot_table(index='Question', columns='Response',
    aggfunc=len, fill_value=0)

# Create a heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(response_pivot, annot=True, cmap="YlGnBu", fmt="d")
plt.title("Likert Scale Questionnaire Responses (3 Participants)")
plt.ylabel("Question")
plt.xlabel("Response")
plt.show()
```

[8] https://en.wikipedia.org/wiki/Ecological_validity

[9] described in page 3 of https://www.ulster.ac.uk/__data/assets/pdf_file/0009/478809/Chatbot-Usability-Questionnaire.pdf

[10] https://en.wikipedia.org/wiki/Heat_map

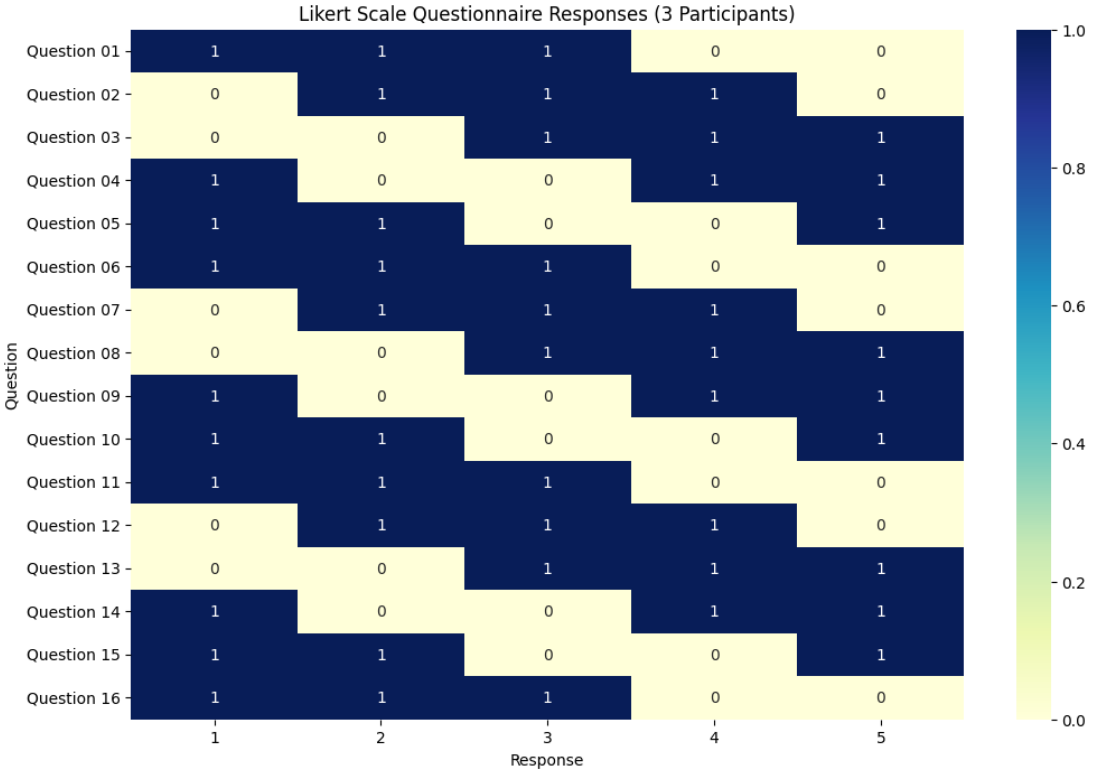

Figure 2: Heatmap visualisation of Likert scale responses

This code should produce the visualisation you can see in Figure 2. You can quickly see from this visualisation where the bulk of the responses are, as they are shaded darker. Note that the given example uses only three participants.

3.3.2 Visualisation through stacked bar charts

Stacked bar charts[11] are a good way to visualise the proportions of a variable under different conditions. Because our questions are on a Likert scale, they are therefore useful to quickly see the overall attitude of the participants through different questions. For example, in Figure 3, we can observe that participants overall agree with the statements of Questions 1 to 5 (majority yellow and light green).

```
import pandas as pd
import matplotlib.pyplot as plt

# Sample data: Responses to 5 questions on a Likert scale from 1 to 5
data = {
    'Strongly Disagree': [5, 2, 3, 4, 1],
    'Disagree': [10, 5, 8, 6, 2],
    'Neutral': [15, 18, 12, 14, 15],
    'Agree': [30, 25, 35, 40, 33],
    'Strongly Agree': [40, 50, 42, 36, 49]
}

df = pd.DataFrame(data, index=[f'Question {i+1}' for i in range(5)])

```

[11] https://en.wikipedia.org/wiki/Bar_chart

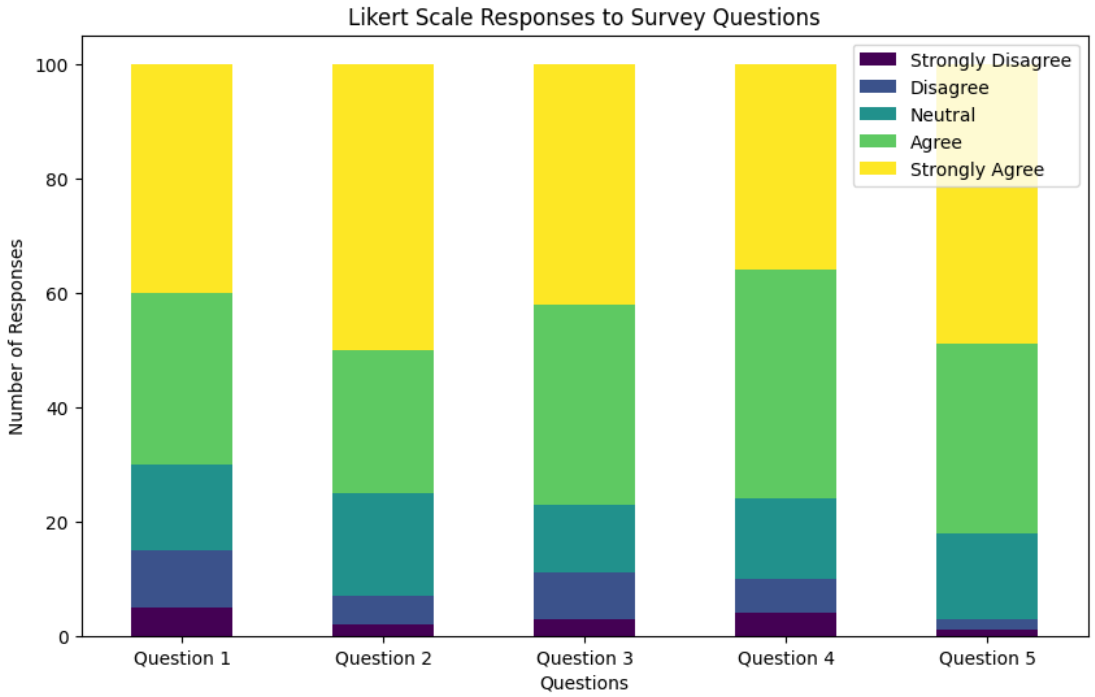

Figure 3: Stacked bar chart visualisation of Likert scale responses

```
# Create a stacked bar chart
ax = df.plot(kind='bar', stacked=True, figsize=(10, 6), colormap='viridis')

# Adding labels and title
plt.xlabel('Questions')
plt.ylabel('Number of Responses')
plt.title('Likert Scale Responses to Survey Questions')
plt.xticks(rotation=0)

# Display the chart
plt.show()
```

The code above should produce a visualisation such as the one in Figure 3, which allows you to quickly see the overall distribution of response per question (on a small sample of 5 questions to make it fit in the page).

Of course, those are not the only ways to visualise responses to a Likert scale survey. In fact, they suffer from (at least) one critical flaw. What is it and how would you fix it?

3.4 Activity

For the purpose of your work in COMP3074, you only have access to a restricted set of participants. This should not stop you from conducting a usability experiment, but instead it should lead you to think about the consequences of generalising from this population. Is that population representative of your target population? What demographics could you be missing? How much of an impact could it have?

3.5 Other visualisations and rules of thumb

Data visualisation is as much an art as it is a science, but there are some general rules of thumb and principles which have been collected over the years by experts such as Edward Tufte[12], a pioneer of data visualisation. While the principles and rules are interesting, here are the ones I would ask you to abide to for your report:

- Keep your plots simple.
- Keep your plot design simple and un-pie-chart-like (because pie charts are bad).
- Do not present the same information twice unless it is necessary (and two visualisations bring different facets of the information to light).
- Sometimes a table with numbers can do the same job as a visualisation and better.
- When using a shaded region to represent a numerical value, the area of that shaded region should be directly proportional to the corresponding value (principle of proportional ink[13]).
- Never use pie charts. They are terrible. Doughnut charts are just as bad.

4 Evaluating RRI

Finally, we have seen how to evaluate systems and how to evaluate user experience. The last thing we need to see is how to evaluate the (potential) impact on the wider world. The field of Responsible Research and Innovation[14] (RRI) focuses on thinking ahead and trying to see the consequences of a system before it is deployed. For the sake of COMP3074 we will take a simplified view of RRI.

4.1 Principles of RRI

RRI is a complex and deep field of applied ethics that would not fit in a single lab of a few hours of work, so we will focus on two particularly important principles of RRI: anticipation (of positive and negative consequences) and reflection (on the innovation process). The goal of those two processes is to dig for insights on how to make the product (here your conversational AI) better, safer, and more inclusive.

4.1.1 Anticipating consequences

The Orbit website defines anticipation as follows[15]:

> Thinking about the possible consequences of research and innovation is a key component of RRI. Concern over the role of science and technology in traumatic developments such as the industrialised mass killings in modern warfare and the potential annihilation of humanity through nuclear weapons led to new ways of reflecting on science and technology research and innovation. Chernobyl and Bhopal have become synonymous with industrial accidents made possible by scientific and technical progress.
>
> It is not only catastrophic one-off events that need to be considered. Current discussions of the social consequences of social media use, for example, focus on the

[12] https://en.wikipedia.org/wiki/Edward_Tufte
[13] https://callingbullshit.org/tools/tools_proportional_ink.html
[14] https://www.orbit-rri.org/about/about-rri/
[15] https://www.orbit-rri.org/resources/keys-of-rri/#anticipation

way in which novel technologies have cumulative effects that were unforeseen and unplanned when these technologies were developed.

This points to a key challenge of anticipation: the future is unknown. How can an individual researcher, research institute or funder be expected to know what all the future consequences of their work will be? Although not all the consequences can be known, we may have strong reasons to believe that some aspects of the future can be successfully anticipated. Past experience can to some degree be extrapolated to expectations of the future. In addition, researchers have expectations about the outcomes of their work. In the UK funding environment, most proposals have to be accompanied by a description of expected 'pathways to impact'. Anticipation in the RRI sense of the word can be more or less formal and elaborated. The key to anticipation for RRI is to ensure that consequences of undertaking the research and of possible findings are considered and that these considerations are reflected in the research design. (from the Orbit website)

Pair yourself up with a classmate (or two) and engage in an RRI anticipation exercise, constructively criticising each other's proposed system using the following guiding questions:

- What are the intended and unintended consequences of this AI?
- What are the positive consequences on which we want to focus?
- What are the consequences that we would ideally want to mitigate?
- How would you mitigate them?[a]

[a]Hint: quite often, and as a recurring theme in COMP3074, engaging with stakeholders is a key step in anticipating and mitigating negative consequences. It is also why Engage is one of the key principles of the RRI AREA framework as detailed in `https://www.ukri.org/who-we-are/epsrc/our-policies-and-standards/framework-for-responsible-innovation/`

4.1.2 Reflecting

The Orbit website defines reflection as follows[16]:

Reflection is at the heart of RRI. Researchers always reflect on the research question they ask, the type of data they collect, the way they analyse the data and the implications of findings. Reflection in the RRI sense goes further to examine the research or innovation activities more broadly. This means asking fundamental questions such as "Is this research required at all?" "Will the (foreseeable) negative consequences of this work be proportional to the (intended) consequences?" or "Could the research question / problem be addressed in a completely different way?"

One way of describing reflection in RRI is to see it as an example of second order reflexivity, i.e. of a reflection on the processes of reflection that underpin and guide research. This means that the axioms and basic assumptions need to be questioned with a view to ensuring that the research is aligned with societal needs and requirements.

Reflection can be an individual activity, with the individual innovator thinking about their work. However, in many cases this will not lead to relevant insights, as it is difficult for most people to clearly understand their own biases and preconceptions. Reflexivity is therefore often best undertaken collectively. It typically forms part

[16]`https://www.orbit-rri.org/resources/keys-of-rri/#reflection`

of engagement activities. There are also many ways of forcing reflection through organisational processes and structures, e.g. in the form of a project advisory board, stage gating processes, review and quality assurance steps and many others. The key is to ensure that there is space in the research activities to take a step back and look at the work from different perspectives. (from the Orbit website)

Pair yourself up with a classmate (or two) and engage in an RRI reflection exercise, constructively criticising each other's proposed system using the following guiding questions:

- What is the point of this system? What does it bring?
- Are the benefits of the system outweighed by the potential negatives?
- Where did the data come from, was it collected ethically?
- Is the system perpetuating some form of bias?

4.2 Activity

Reread the results from the anticipation and reflection activities you just did and think about how you will integrate that in your final coursework.

References

[1] Tukur Dahiru. P-value, a true test of statistical significance? a cautionary note. *Annals of Ibadan postgraduate medicine*, 6(1):21–26, 2008.