Mark Klaiman — 208734715 | Omri Gruman — 318965621

Analysis of Unrecognized User Requests in Goal-Oriented Dialog Systems

1. Introduction

In this project, we were given the task of analyzing unrecognized user requests taken from goal-oriented dialog systems. We had to start by clustering the entire batch of requests in order to reveal different sub-topics. The analysis continues by extracting representatives from each cluster to identify the diversity of the cluster. We finish our work by trying to construct a meaningful name to each of the clusters.

2. Request clustering

To surface topical clusters in unhandled requests, we had to come up with a solution to satisfy two important conditions – the number of clusters was unknown, and outliers should be avoided. For this task we proposed few solutions such as X-Means, OPTICS, DBSCAN, and Community Detection. DBSCAN performed best until we finally found the Community Detection method implemented by the sentence-transformers library.

2.1. Sentence embeddings

To apply a clustering algorithm on the requests, we first encoded each user request using an SBERT model, specifically *all-MiniLM-L6-v2*. This resulted in a high-dimensional vector representation of each request.

2.2. Community detection

We fed the request embeddings to the *community_detection* function, along with a few parameters that controlled the minimum cluster size and the similarity threshold. The output of the *community_detection* function was a list of clusters, where each cluster was represented by a list of indices corresponding to the original user requests. The given list was then converted to a list of cluster id’s, corresponding to the given list of requests.

2.3. Evaluation

To evaluate the clustering outcome, we used the provided solution and computed the Rand Index (RI) and Adjusted Rand Index (ARI) using the compare_clustering_solutions.py utility script that was provided to us in advance.

Using Community Detection resulted in the following scores:

Dataset 1:

- **30 clusters**
- **466 unclustered requests**
- **RI = 0.915**, **ARI = 0.639**

Dataset 2:

- **212 clusters**
- **2627 unclustered requests**
- **RI = 0.949**, **ARI = 0.533**

Indicating that the clustering outcome was reasonably close to the provided solution.

2.4. Results

To evaluate the quality of the chosen clustering method we also examined the clusters ourselves, as seen below:

Example:

In the provided solution, the request:

(1) "what exactly is home quarantine?"

was clustered with the requests:

(2) "how long do i have to stay in quarantine?"
(3) "what if i live by myself and i'm in quarantine?"
(4) "can someone come see me during quarantine?"

Analysis:

We analyzed the clustering results by examining their similarity to the provided solution. We found that the clusters contained requests similar to the provided solution, indicating that the community detection algorithm is an effective method for clustering user requests.

In our solution, requests (2) and (3) were in the same cluster as request (1), but request (4) was left out and labels as an outlier.
This shows that the *community_detection* clustering algorithm is not perfect.

3. Cluster representative extraction

After clustering our requests, we needed to find a way to find the most representative requests in each cluster. Nevertheless, our representatives had to be different enough to showcase the diversity of the cluster. For that matter, we introduced two main approaches, the first one involving dimensionality reduction with PCA, and the other one using the K-Means++ initialization method. Finally, we chose to apply the K-Means++ method over the PCA method.

3.1. K-Means++

K-means++ is a modification of the original K-means algorithm and it was proposed to improve the quality of the initial cluster centroids. The main idea behind K-means++ is to select the initial centroids in a way that maximizes the chances of finding good cluster assignments.

I. The first centroid is chosen uniformly at random from the data points.
II. For each data point, the distance to the nearest centroid that has already been chosen is calculated.
III. The next centroid is chosen randomly from the data points, with the probability of each point being proportional to the squared distance to the nearest centroid.
IV. Steps (II) and (III) are repeated until all K centroids have been chosen.

By selecting the representatives in this way, K-means++ ensures that they are well spread out across the data space and cover the variation in the cluster.

3.2. Results

Given the following requests that were clustered together, the ones that were chosen as representatives are marked:

- "***can i return to the united states?***",

"returning from abroad",
"i am in europe and i want to return to the united states. is that possible?",

- "***returning from abroad***",

"can i return to the united states if i am abroad?",
"returning from abroad",
"returning from abroad",
"returning from abroad",
"returning from abroad",

- "***can i travel out of the country?***",

"when can i return to the united states?"

4. Cluster Naming

The final part of our project included giving each cluster a meaningful name. That is, a name that holds the main topic of the cluster and is also fluent. The approach that we suggested is looking for an N-gram within the cluster that would be the most representative of the cluster main idea. In order to rank the N-grams that we found, we had to come up with a scoring technique that would allow us to select the best N-gram in the cluster.

4.1. N-gram scores

As preprocessing for the scoring stage, we counted the appearances of individual words that are not stopwords to measure the importance of individual words in the cluster. Next, we counted the appearances of N-grams between 2 and 4 words within the cluster.

Then, for each N-gram, we assigned a score based on the following factors:

1. N-gram length (+1 to balance the 4th factor)
2. Number of appearances of the N-gram in the cluster
3. The number of appearances of the individual non-stopwords in the N-gram
4. The amount of stopwords in the N-gram (+1 for 0-division)

To compute the score, we multiplied the first three factors and divided them by the fourth factor. The resulting score was used to assign a label to each cluster.

We found that this approach was effective in generating descriptive and relevant cluster names. The use of N-grams allowed for capturing more complex and meaningful phrases, while the incorporation of stopwords helped to focus on the most meaningful words as an indicator for an important phrase. However, we noted that the effectiveness of this approach is highly dependent on the quality and quantity of data used for clustering. Moreover, the use of N-grams alone sometimes resulted in phrases that were not perfectly fluent, but the main idea remained.

Here's a short example of how the factors would be calculated for the N-gram "*data and science*" in a cluster with the following information:

I. N-gram length: 3
II. Number of appearances of the N-gram in cluster: 7
III. Number of appearances of the individual non-stopwords in N-gram:
    - "*data*": 10
    - "*science*": 7
IV. Amount of stop words in N-gram: 1

Calculation:

Factor (I):	3 + 1 = 4
Factor (II):	7
Factor (II):	10 x 7 = 70
Factor (IV):	1 + 1 = 2
Final score:	(4 x 7 x 70) / 2 = **980**

4.2. Results

Given the following requests that were clustered together, the name that was constructed for this cluster is “prevent the spread”.

"how can i lower my temperature if i have the coronavirus?",
"how can i protect my family from the spread of coronavirus?",
"what is the best prevention?",
"what medicines should i take if i have coronavirus",
"what is the best way to prevent coronavirus",
"what medicine should i avoid taking if i have coronavirus",
"how often do i need to clean my home to prevent the spread of coronavirus?",
"prevent the spread of the virus",
"what is the best prevention?",
"how to prevent the spread of the virus",
"prevent the spread of the virus"

5. Conclusion

In this project we successfully tackled a real-world unsupervised problem. We were able to cluster the unhandled requests to clusters using only the request sentences, extract some informative instances that would help us understand the variation in each cluster, and finally we named the different clusters to label their main ideas.