המכללה האקדמית תל־אביב־יפו

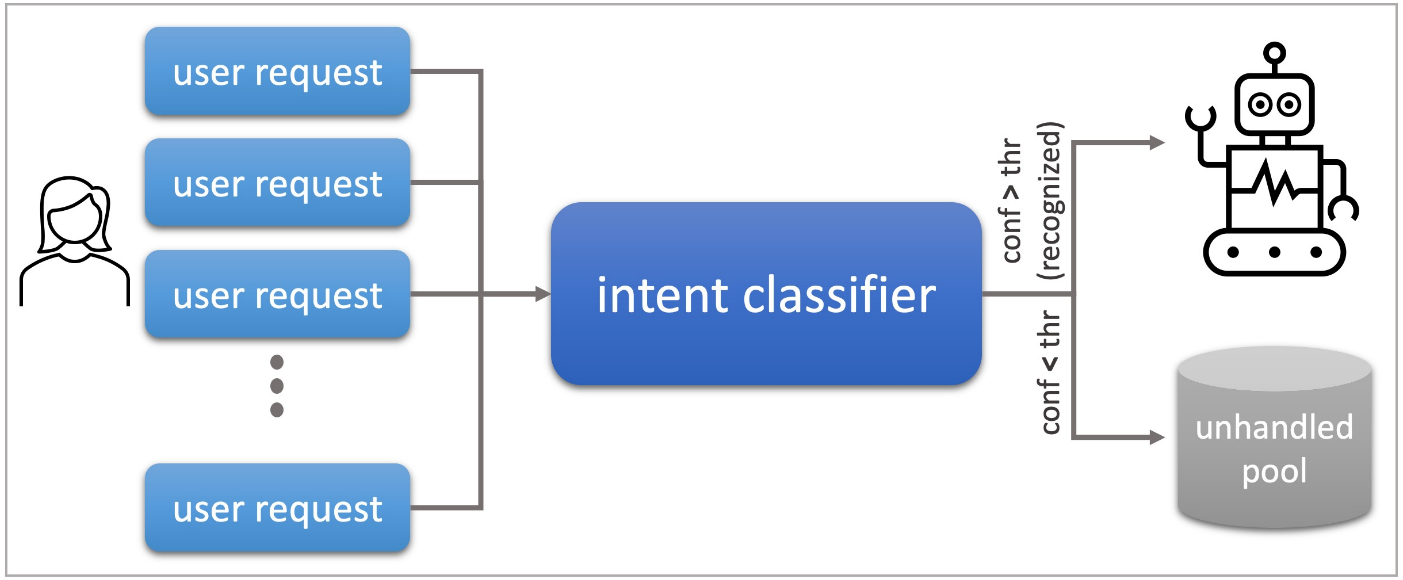
Analysis of Unrecognized User Requests in Goal-Oriented Dialog Systems

the final project

submission date: 25.03.2024

MOTIVATION AND BACKGROUND

- goal-oriented dialog systems, a.k.a virtual assistants (VAs), often fail to recognize the intent of natural language requests
- in practice, these cases are normally identified using intent classifier uncertainty – requests that are predicted to have a level of confidence below a certain threshold are reported as unrecognized (or unhandled)

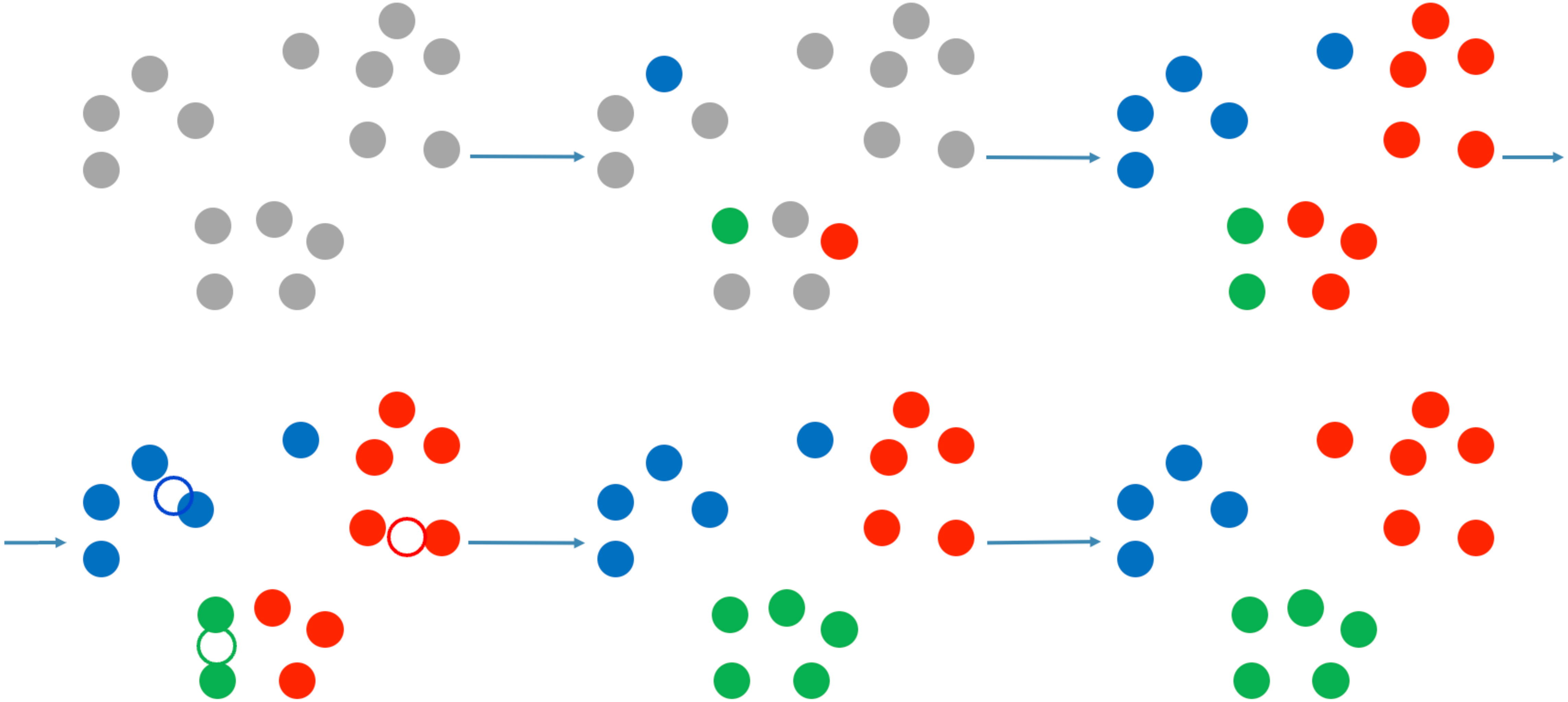


MOTIVATION AND BACKGROUND

- unhandled requests carry over various aspects of potential importance
  - novel examples of existing intents, completely novel topics, seasonal peaks
- in large deployments the number of unhandled requests can reach tens of thousands daily, making manual inspection impractical

- our goal is to propose (and implement) an approach for
- (1) surfacing topical clusters in unhandled requests (clustering)
- (2) cluster naming (labeling)

CLUSTERING – K-MEANS (reminder)

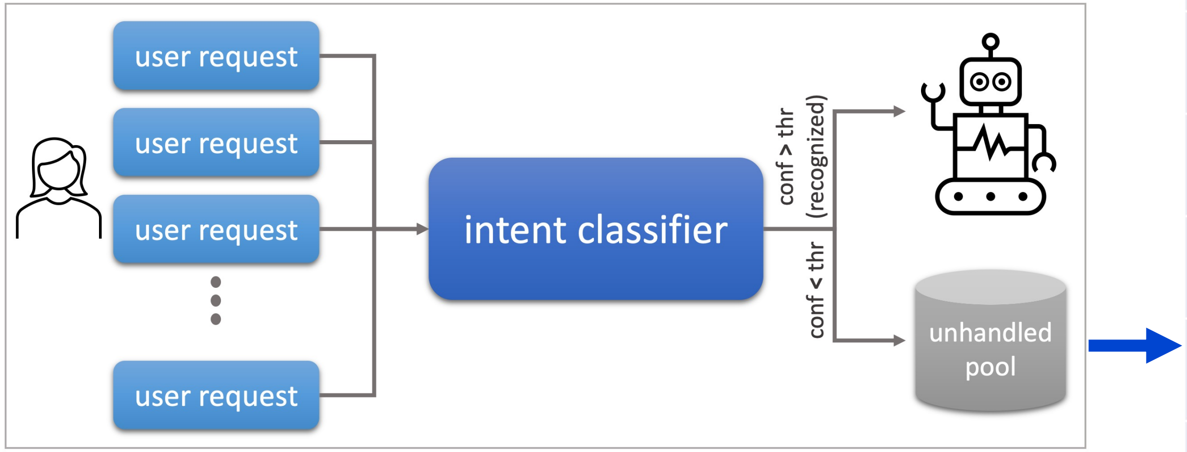


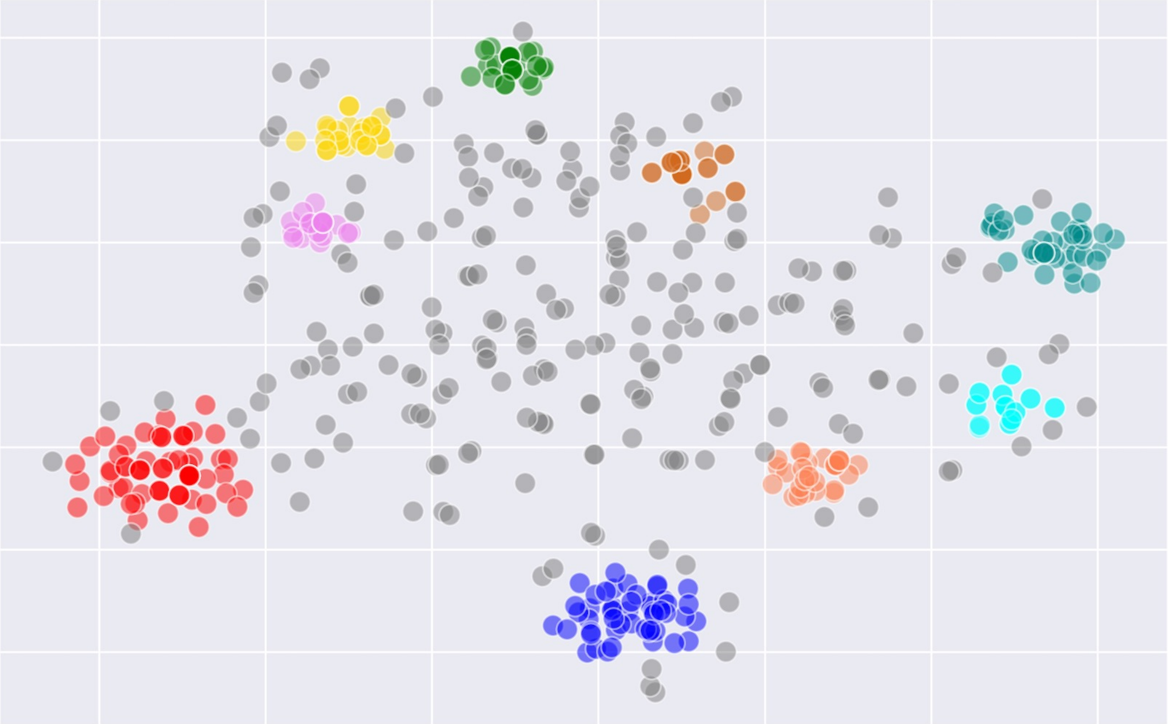
CLUSTERING – K-MEANS (reminder)

```
## K-Means clustering

K = the number of clusters
place cluster centroids $c_1, c_2, … , c_k$ randomly
repeat till convergence or till the end of a fixed number of iterations
        for each data point $x_i$
                find the nearest centroid ($c_1, c_2, … , c_k$)
                assign the point to that cluster
        # end for
        for each cluster j=1..k
                new centroid = mean of all points assigned to that cluster
        # end for
# end repeat
```

(1) CLUSTERING REQUESTS





what properties should the clustering approach satisfy?

(1) CLUSTERING REQUESTS – requirements

Clustering solutions can be roughly categorized into across two major dimensions:

✗ *requiring a predefined (fixed) number of output clusters* vs *discovering the number of clusters as part of the clustering algorithm* ✓

✗ *forcing cluster assignment on the entire dataset* vs *tolerating outliers* ✓

(the K-MEANS clustering)

required in our use-case

(1) CLUSTERING REQUESTS – example

cluster name: **difference covid flu** (28)	cluster name: **covid during pregnancy** (17)
is covid the same as flu? (4)	covid 19 and pregnancy (6)
how is covid different from the flu? (3)	covid risk for a pregnant woman (4)
what’s the difference between covid 19 and flu?	what is the risk of covid for pregnant women?
what’s the difference between covid and flu	is covid-19 dangerous when pregnant?
is the covid the same as cold?	7 months pregnant and tested positive for covid, any risks?
covid vs flu vs sars	covid 19 during pregnancy
…	…

(1) CLUSTERING REQUESTS – algorithm suggestion

- encode a set of m unhandled requests: R=(r1, r2, r3, …) into their vector representations (embeddings) E=(e1, e2, e3, …) using an LLM encoder
- iterate over representations in E, where each request can be assigned to an existing cluster (if its proximity to the cluster's centroid meets some similarity threshold), otherwise the request initiates its own cluster
- additional iterations over all request embeddings are performed till the algo convergence or till the max number of iterations is exhausted
- clusters with size exceeding a pre-defined min_size are reported as generated clusters
    - all other requests are considered unclustered

(1) CLUSTERING REQUESTS – use this encoder

```
from sentence_transformers import SentenceTransformer

# full documentation - https://huggingface.co/sentence-transformers
MODEL_NAME = 'all-MiniLM-L6-v2’

model = SentenceTransformer(MODEL_NAME)
```

EVALUATION

- (1) surfacing topical clusters in unhandled requests (clustering)
- multiple dataset(s) with (good) clustering solution will be provided as a ground truth
- quantitative evaluation

- (2) cluster naming (labeling)
- example assignment of cluster names will be provided
- qualitative evaluation

CLUSTERING EVALUATION: GROUND TRUTH IS KNOWN

- rand index (RI) and adjusted rand index (adjusted RI)
- the number of un-clustered instances
- the number of clusters

PROJECT SUBMISSION – a single zip file with

- a report (2-3 pages) including the description of
  - your approach to the task (the two parts)
  - evaluation of the clustering outcome against the provided solution – RI, ARI
    - use the provided `compare_clustering_solutions.py` to compute these scores
  - any essential details about running your code (e.g., anticipated runtime)
- your outcome on the datasets attached to the project – two output json files
  - the output files should be precisely in the same json format as the provided solutions
- your code (the `main.py` file)

* implement your own clustering code, do not make use of existing solutions