nlu_rag_eda_synth

November 11, 2025

[ ]:

1 NLU and RAG Templates EDA

This notebook inspects the synthetic NLU dataset (user queries) and the RAG template corpus used for retrieval. It includes tokenization examples, label distributions, synonym augmentation, and simple template matching checks.

[2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
nlu = pd.read_csv('./nlu_dataset.csv')
print('NLU data sample:')
nlu.head()
```

```
NLU data sample:
```

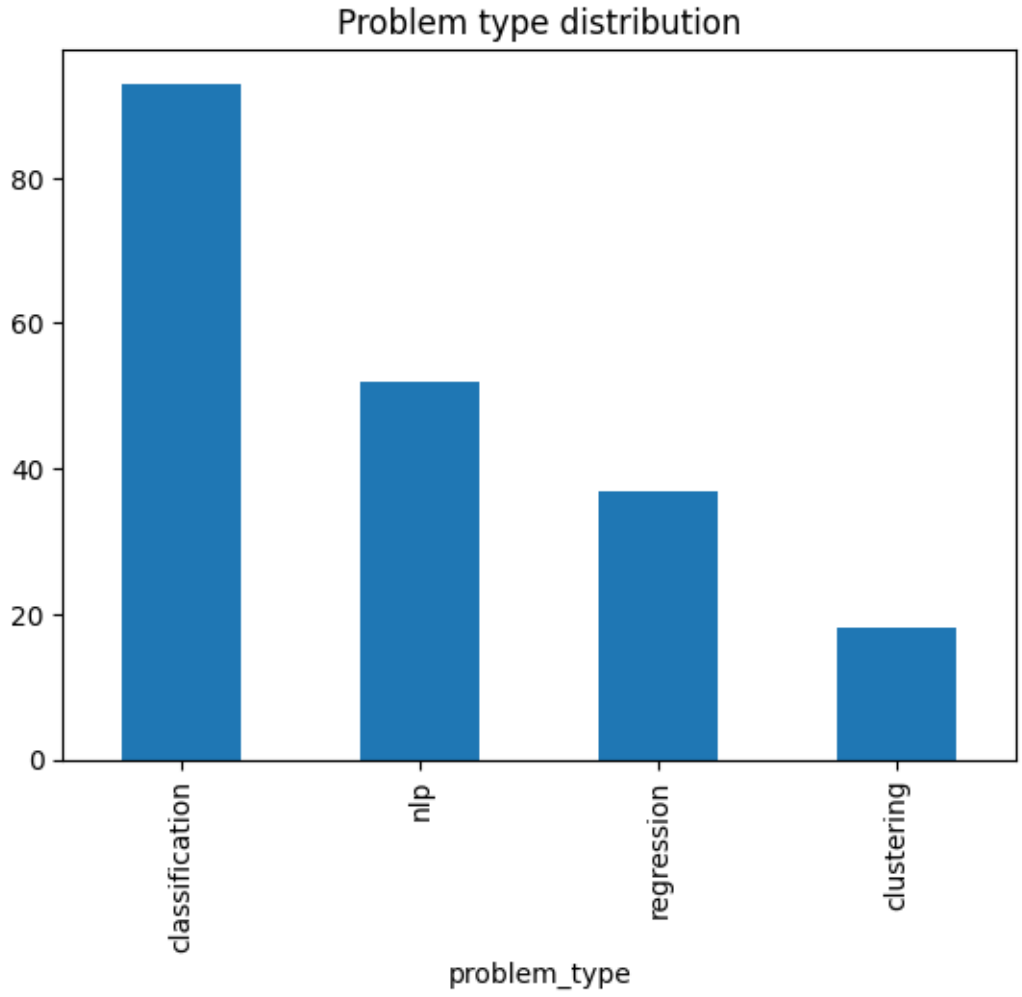
[2]:

	user_query	problem_type	domain	target_variable	algorithm_preference
0	Create a voice activity detector for audio rec…	classification	tabular	NaN	NaN
1	Create a voice activity detector for audio rec…	classification	tabular	NaN	NaN
2	Create a voice activity detector for audio rec…	classification	tabular	NaN	NaN
3	Cluster customers into segments based on purch…	clustering	tabular	NaN	NaN
4	Build a supervised classification model to pre…	classification	tabular	churn	NaN

1.1 Label distribution

[3]:
```
print(nlu['problem_type'].value_counts())
nlu['problem_type'].value_counts().plot(kind='bar')
plt.title('Problem type distribution')
plt.show()
```

```
problem_type
classification    93
nlp               52
regression        37
clustering        18
Name: count, dtype: int64
```



1.2 Tokenization and keyword density

[4]:
```
import re
nlu['token_count'] = nlu['user_query'].apply(lambda x: len(re.findall(r"\w+",␣
 ↪x)))
keywords =␣
 ↪['predict','classify','regress','image','text','cluster','summarize','detect','forecast']
def keyword_density(s):
    s2 = s.lower()
```

```
    return sum(s2.count(k) for k in keywords)/max(1,len(s2.split()))
nlu['keyword_density'] = nlu['user_query'].apply(keyword_density)
nlu[['token_count','keyword_density']].describe()
```

[4]:

	token_count	keyword_density
count	200.000000	200.000000
mean	9.055000	0.121661
std	2.758627	0.081902
min	6.000000	0.000000
25%	7.000000	0.076923
50%	8.000000	0.125000
75%	12.000000	0.166667
max	15.000000	0.285714

1.3 RAG templates preview

[6]:
```
import json
templates = []
with open('./rag_templates.jsonl','r') as f:
    for line in f:
        templates.append(json.loads(line))
print('Templates loaded:', len(templates))
for t in templates:
    print('-', t['template_id'], t['task_type'], t['framework'])
```

```
Templates loaded: 5
- tpl_0 classification scikit-learn
- tpl_1 regression scikit-learn
- tpl_2 nlp sklearn
- tpl_3 vision pytorch
- tpl_4 time-series statsmodels
```

1.4 Simple matching test (keyword -> template task)

[7]:
```
def match_template(query):
    q = query.lower()
    if 'image' in q or 'cat' in q or 'dog' in q:
        return 'vision'
    if 'price' in q or 'forecast' in q or 'regress' in q:
        return 'regression'
    if 'review' in q or 'text' in q or 'sentiment' in q:
        return 'nlp'
    return 'classification'

nlu['matched_task'] = nlu['user_query'].apply(match_template)
import pandas as pd
print(pd.crosstab(nlu['problem_type'], nlu['matched_task']))
```

matched_task	classification	nlp	regression	vision
problem_type				
classification	28	16	5	44
clustering	18	0	0	0
nlp	39	5	8	0
regression	10	8	19	0

1.5 Observations

- NLU intents are extractable with keyword heuristics, but an LLM-based NLU will improve robustness.
- RAG templates are small; for production, enlarge corpus and index embeddings.

[ ]:

[ ]: