IRSE Projecet

Jan Cichomski (r1026448)

May 2, 2025

Outline

1. Architecture

TODO

2. Term Vocabulary

2.1 Term Vocabulary - Document Preprocessing

- To lower case
- remove punctuation
- tokenize
- remove english stop words (added custom stop words)
- lammatize

2.1 Temr Vocabulary - Document Preprocessing

```
def preprocess_text(doc):
    doc = doc.translate(str.maketrans("", "",
        string.punctuation)).lower()
    words = word_tokenize(doc)
    words = [
        lemmatizer.lemmatize(word)
        for word in words
        if word not in stop_words and word.isalpha()
    ]
    return " ".join(words)
```

2.1 Term Vocabulary - Custom stop words

```
stop_words.update(
    [
        "add",
        "added",
        "adding",
        "addition",
        "also",
        "almost",
        "another",
        "easily",
        "easy",
    ]
)
```

2.2 Term Vocabulary - Hyperparameters

Two types of terms:

- 1-grams
  - min_df=20
  - max_df=0.5
- 2-grams
  - 10,000 terms
  - min_df=50
  - max_df=0.4

2.3 Term Vocabulary - Handling mulit-word terms

- 2-grams with aggressive filtering

3 Document Embedding

3.1 Document Embedding - Chosen Fields

I use all fields for embedding:

- name
- description
- ingredients
- steps

- Tested different combinations
- Make sense as user may ask about any information

TODO: add some data

3.2 Document Embedding - Query Preprocessing

The same approche as for embedding documents

```
def retrieve_documents(query_text, recipies,
                  recipe_ids, k, threshold):
query = preprocess_text(query_text)
...
```

3.3 Document Embedding - Edge Cases

- Problem: When query has no terms from vocabulary
    - TF-IDF produces zero vector for the query
    - Cosine similarity returns 0 for all documents
- Consequences:
    - Without similarity threshold: All documents returned (no filtering)
    - With any similarity threshold: No documents returned (empty result)

4 Retrieval

4.1 Retrieval - Similarity Measure

- Cosine similarity - picked finally
- Euclidean distance

4.2 Retrieval - Hyperparameters

- Max number of returned documents: 40
- Minimum threshold for cosine similarity: 0.2

I used grid search over param space

```
def create_parameter_heatmap(queries, recipes, recipe_ids):
  thresholds = np.arange(0.1, 0.60, 0.05)
  k_values = np.arange(20, 60, 5)
```

4.3 Retrieval - Evaluation Metrics

- Macro Precision: 0.130
- Macro Recall: 0.201
- Macro F1: 0.126
- Micro Precision: 0.128
- Micro Recall: 0.191
- Micro F1: 0.153

4.4 Retrieval - MAP

Mean Average Precision (MAP): 0.086

$$AP = \frac{1}{RD} \sum_{k=1}^{n} P(k) \cdot r(k), \tag{1}$$

Were RD is the number of relevant documents for the query, n is the total number of documents, $P(k)$ is the precision at k, and $r(k)$ is the relevance of the k^{th} retrieved document (0 if not relevant, and 1 if relevant)

$$MAP = \frac{1}{Q} \sum_{i=1}^{Q} AP_i \tag{2}$$

Where Q is the number of queries and AP_i is the average precision for the i^{th} query.

4.4 Retrieval - MAP Code

```
def calculate_average_precision(relevant_doc_ids,
                                retrieved_doc_ids):
  hit_count = 0
  sum_precisions = 0.0
  for i, doc_id in enumerate(retrieved_doc_ids):
      if doc_id in relevant_doc_ids:
          hit_count += 1
          precision_at_i = hit_count / (i + 1)
          sum_precisions += precision_at_i
      # else: sum_precisions += 0.0
  if len(relevant_doc_ids) == 0:
      return 0.0
  return sum_precisions / len(relevant_doc_ids)
```

5. Qualitative analysis - information Retrieval

TIODO TODO TODO

5.1 Qualitative analysis - IR -

Problem: Even though there is no relevant information in the document, the system returns somed documents
Prompt: "Where can I follow cooking classes"
Output: MB in hand outs???

5.2 Qualitative analysis - IR -

Problem: Ignores context of entities in query
Prompt: ”How does Gordon Ramsay make his beef Wellington?”
Output: MB in hand outs???

5.3 Qualitative analysis - IR -

Problem: Can't handle extermalyl rare words, like "Paraguay"
Prompt: "Do you know any soups from Paraguay?"
Output: MB in hand outs???

5.4 Qualitative analysis - IR -

Problem: TF-IDF doesn't handle typos
Prompt: "How do you make **piza**"
Output: MB in hand outs???

5.5 Qualitative analysis - IR -

Problem: Can't capture negation
Prompt: "I do not want to eat pizza, what can I eat instead?"
Output: MB in hand outs???

6. Prompt

6.1 Prompt - LLM Instructions - Good

- General context and LLM's goal
- Instructions per kind of question
- Response format
- Limitations

TODO: add full prompt in handout

6.1 Prompt - LLM Instructions - Bad

```
You are a helpful recipe assistant with access to a database
of recipes. The system has already retrieved the most
relevant recipes to the user's query using TF-IDF similarity.
Your goal is to provide helpful,accurate responses about
recipes, cooking techniques, ingredient substitutions, and
culinary advice based on the retrieved recipes.

The following recipes have been retrieved as most relevant
to the user's query:
{retrieved_recipes}

## User Query
{user_query}
```

6.2 Prompt - Fields used

- name
- description
- ingredients
- steps
- relevance score

6.2 Prompt - Fields used

```
for idx, (recipe, recipe_id, score) in enumerate(results):
  info=df[df["official_id"] == recipe_id].iloc[0]
  retrieved_recipes+=f"Document {idx}, Score: {score:.4f}\n"
  retrieved_recipes+=f"Name: {info[’name’]}\n"
  retrieved_recipes+=f"Description:{info[’description’]}\n"
  retrieved_recipes+=f"Ingredients:{info[’ingredients’]}\n"
  retrieved_recipes+=f"Steps: {info[’steps’]}\n\n"
```

7. Qualitative analysis - LLM

7.1 Qualitative analysis - LLM & RAG

- more or less yes
- but does not stick to rules
- they are often too general

7.2 Qualitative analysis - TODO

7.3 Qualitative analysis - Hallucination

- Standard prompt, but with no documented provided
- Yet LLM answered
- Which suggest that it did not followed rules

7.4 Qualitative analysis - Score vs No Score - TODO

8. Neural document embeddings

8.1 Neural document embeddings - out of vocabulary - query

Used model's tokenizer to tokenize the query

- Original text: 'kashubian'
- `[CLS]', 'ka', '##shu', '##bian', '[SEP]'`

The ## prefix represents subword tokenization, allowing the model to handle words not in its vocabulary.

8.1 Neural document embeddings - out of vocabulary - document

Used model's tokenizer to tokenize the query

- Original text: 'their settlement area is referred to as kashubia they speak the kashubian language which is classified either...'
- `'[CLS]', 'their', 'settlement', 'area', 'is', 'referred', 'to', 'as', 'ka', '##shu', '##bia', 'they', 'speak', 'the', 'ka', '##shu', '##bian', 'language', 'which', 'is', 'classified', 'either',`

Neural embeddings returned valid results even through the word "kashubian" was not in the vocabulary.

8.2 Neural document embeddings - Metrics - Recipes

- Macro Precision: 0.307
- Macro Recall: 0.134
- Macro F1: 0.157
- Micro Precision: 0.280
- Micro Recall: 0.035
- Micro F1: 0.062
- MAP: 0.101
- Average DCG: 2.472
- Average NDCG: 0.736

8.2 Neural document embeddings - Metrics - Wiki

- Macro Precision: 0.310
- Macro Recall: 0.352
- Macro F1: 0.260
- Micro Precision: 0.343
- Micro Recall: 0.303
- Micro F1: 0.322
- MAP: 0.216
- Average DCG: 1.566
- Average NDCG: 0.549

9. Compression

9.1 Compression - Long Dcouments

Information Retrieval

- Cover multiple topics
- Contain lots of words
- Limited document length

LLM

- Limited context widnow
- Needle in a haystack

9.2 Compression - Solutoin

Split documents into chunks

```
from langchain_text_splitters
  import RecursiveCharacterTextSplitter
text_splitter = RecursiveCharacterTextSplitter(
  chunk_size=500,
  chunk_overlap=100,
  length_function=len,
  is_separator_regex=False,
)
```

9. Security

9.1 Security

- Yes, LLM is susectipble TODO...