WSM-Project3

109703040洪畯宸
109703054 莊維軒

Slide here!

TOC

1. TFIDF
2. ItemCF
3. N-Gram
4. Discussions & Feedbacks

TFIDF

TFIDF

- Port project1's code

```
# Build dimension dictionary, indicates key's index
index = 0
for word in self.parser.parse(" ".join([doc["doc"] for doc in docs]), language):
  if word not in self.dimensions:
    self.dimensions[word] = index
    index += 1
self.tf = np.zeros((len(docs), len(self.dimensions)))

# Count tf, df, idf, and tfidf by using numpy functions, and store them for further u
for i, doc in enumerate(tqdm.tqdm(docs, postfix="Building Index")):
  processedDoc = self.parser.parse(doc["doc"], language)
  doc["parsed"] = processedDoc
  for word in processedDoc:
    self.tf[i][self.dimensions[word]] += 1
```

TFIDF

- Port project1's code

```
# Build dimension dictionary, indicates key's index
index = 0
for word in self.parser.parse(" ".join([doc["doc"] for doc in docs]), language):
  if word not in self.dimensions:
    self.dimensions[word] = index
    index += 1
self.tf = np.zeros((len(docs), len(self.dimensions)))

# Count tf, df, idf, and tfidf by using numpy functions, and store them for further u
for i, doc in enumerate(tqdm.tqdm(docs, postfix="Building Index")):
  processedDoc = self.parser.parse(doc["doc"], language)
  doc["parsed"] = processedDoc
  for word in processedDoc:
    self.tf[i][self.dimensions[word]] += 1
```

TFIDF

- Port project1's code

```
# Build dimension dictionary, indicates key's index
index = 0
for word in self.parser.parse(" ".join([doc["doc"] for doc in docs]), language):
  if word not in self.dimensions:
    self.dimensions[word] = index
    index += 1
self.tf = np.zeros((len(docs), len(self.dimensions)))

# Count tf, df, idf, and tfidf by using numpy functions, and store them for further u
for i, doc in enumerate(tqdm.tqdm(docs, postfix="Building Index")):
  processedDoc = self.parser.parse(doc["doc"], language)
  doc["parsed"] = processedDoc
  for word in processedDoc:
    self.tf[i][self.dimensions[word]] += 1
```

14k sessions * 1m songs = 140,000,000,000 `ints` / `shorts` / `bit`

ItemCF

ItemCF

Concept

Predict user preferences based on user's historical behavior, instead of attributes of the item.

Formula

- Basic

$$W_{\mu v} = \frac{| N(\mu) \cap N(v) |}{| N(\mu) |}$$

- Adding penalty term

$$W_{\mu v} = \frac{| N(\mu) \cap N(v) |}{\sqrt{| N(\mu) || N(v) |}}$$

ItemCF

Implementation

ItemCF

Implementation

- Use `dict` to create co-occurance matrix.

	Song1	Song2	…	Song10000
Song1	1000	512	…	201
Song2	512	1	…	345
…	…	…	…	…
Song10000	201	345	…	2000

ItemCF

Implementation

- Use `dict` to create co-occurance matrix.

	Song1	Song2	…	Song10000
Song1	1000	512	…	201
Song2	512	1	…	345
…	…	…	…	…
Song10000	201	345	…	2000

- Do cosine similarity search based on the matrix

ItemCF

Implementation

- Use `dict` to create co-occurance matrix.

	Song1	Song2	…	Song10000
Song1	1000	512	…	201
Song2	512	1	…	345
…	…	…	…	…
Song10000	201	345	…	2000

- Do cosine similarity search based on the matrix
- Randomly fill with top 20 songs if recommendation less than 5 songs.

ItemCF

Implementation

- Use `dict` to create co-occurance matrix.

	Song1	Song2	…	Song10000
Song1	1000	512	…	201
Song2	512	1	…	345
…	…	…	…	…
Song10000	201	345	…	2000

- Do cosine similarity search based on the matrix
- Randomly fill with top 20 songs if recommendation less than 5 songs.
- **Basic Score: 0.14537**

ItemCF

Consider Listening Time

ItemCF

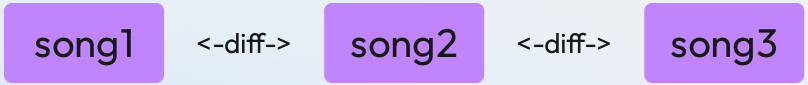
Consider Listening Time

- Give weights based on listening time.

ItemCF

Consider Listening Time

- Give weights based on listening time.
- Use `unix_played_at` to get listening time.



ItemCF

Consider Listening Time

- Give weights based on listening time.
- Use `unix_played_at` to get listening time.

song1 <-diff-> song2 <-diff-> song3

- Our weight:
  - < 10 sec: 0.1
  - 11 ~ 150 sec: 0.7
  - > 150 sec: 1

ItemCF

Consider Listening Time

- Give weights based on listening time.
- Use `unix_played_at` to get listening time.

  song1 <-diff-> song2 <-diff-> song3

- Our weight:
  - < 10 sec: 0.1
  - 11 ~ 150 sec: 0.7
  - > 150 sec: 1
- **Score: 0.10981** (worse)

ItemCF

Consider Repeat Time

ItemCF

Consider Repeat Time

- If last 3 songs are the same song or more than 8 repeated songs among top 20 songs
  - Recommend the same song 5 times

ItemCF

Consider Repeat Time

- If last 3 songs are the same song or more than 8 repeated songs among top 20 songs
  - Recommend the same song 5 times
- **Score: 0.12708** (worse)

ItemCF

Conclusion

ItemCF

Conclusion

Listening Time

- < 10 sec: 0.1, 11 ~ 150 sec: 0.7, > 150 sec: 1 => **Score: 0.10981**
- < 10 sec: 0.1, > 11 sec: 1 => **Score: 0.115**
- Predict what KKbox predicts, instead of what the user would like.

ItemCF

Conclusion

Listening Time

- < 10 sec: 0.1, 11 ~ 150 sec: 0.7, > 150 sec: 1 => **Score: 0.10981**
- < 10 sec: 0.1, > 11 sec: 1 => **Score: 0.115**
- Predict what KKbox predicts, instead of what the user would like.

Repeat Time

- more than 8 repeated songs among top 20 songs => **Score: 0.12708**
- more than 20 repeated songs among top 20 songs => **Score: 0.12755**
- Rating Criteria: Recommend a variety of unique songs get higher scores.

N-Gram

N-Gram

Prepare Data

- Join tables together for furthur usage

```
andyjjrt ~/c/wsm-project3 python main.py                                                    ✔ wsm-project3 Py andyjjrt@nccucs108 master ?8
         session_id    song_id  unix_played_at  ...              lid                                 pid                       title_text_id
0                 1   s_354122      1660012505  ...              NaN                                 NaN  c1079ef109db2aba72f78c632ab73803
1                 1  s_1030665      1660012730  ...              NaN                                 NaN                             NaN
2                 1   s_642781      1660015113  ...         l_440385                            p_288113  f9b7f48dbd07a9979e64ccf88af181aa
3                 1   s_280722      1660015289  ...         l_169111                                 NaN  c1079ef109db2aba72f78c632ab73803
4                 1    s_90294      1660015841  ...         l_196433                            p_192217  9f25d97515a9e19da4c7ad6dba6d8776
...             ...        ...             ...  ...              ...                                 ...                             ...
14306470     715323   s_355651      1664707185  ...              NaN                                 NaN  0025132fbe92679e3472ba869b60af35
14306471     715323   s_170450      1664707201  ...              NaN  p_71934.p_72220.p_72837.p_314936  c1079ef109db2aba72f78c632ab73803
14306472     715323   s_233844      1664707400  ...  l_29174.l_257878                                 NaN  5bb36dbeb12e9e4149eed8166a60fcf8
14306473     715323   s_931390      1664707449  ...         l_284779                                 NaN  859a2ae6d55da7cdfc57bc85d2008a99
14306474     715323   s_935948      1664707668  ...         l_294002                                 NaN  0d42e410deb1c569568ba132e738d6aa
```

N-Gram

Prepare Data

- Join tables together for furthur usage

```
 andyjjrt  ~/c/wsm-project3 python main.py                                    ✔  wsm-project3 Py  andyjjrt@nccucs108  master ?8
          session_id    song_id  unix_played_at  ...               lid                             pid                  title_text_id
0                  1   s_354122      1660012505  ...               NaN                             NaN  c1079ef109db2aba72f78c632ab73803
1                  1  s_1030665      1660012730  ...               NaN                             NaN                               NaN
2                  1   s_642781      1660015113  ...          l_440385                        p_288113  f9b7f48dbd07a9979e64ccf88af181aa
3                  1   s_280722      1660015289  ...          l_169111                             NaN  c1079ef109db2aba72f78c632ab73803
4                  1    s_90294      1660015841  ...          l_196433                        p_192217  9f25d97515a9e19da4c7ad6dba6d8776
...              ...        ...             ...  ...               ...                             ...                               ...
14306470      715323   s_355651      1664707185  ...               NaN                             NaN  0025132fbe92679e3472ba869b60af35
14306471      715323   s_170450      1664707201  ...               NaN  p_71934.p_72220.p_72837.p_314936  c1079ef109db2aba72f78c632ab73803
14306472      715323   s_233844      1664707400  ...  l_29174.l_257878                             NaN  5bb36dbeb12e9e4149eed8166a60fcf8
14306473      715323   s_931390      1664707449  ...          l_284779                             NaN  859a2ae6d55da7cdfc57bc85d2008a99
14306474      715323   s_935948      1664707668  ...          l_294002                             NaN  0d42e410deb1c569568ba132e738d6aa
```

- Trigram

```
n = 3
train_data, padded_sents = padded_everygram_pipeline(n, allWords)
```

N-Gram

model.score

```
result = list(model.score(words[-2:]))
result = [r for r in result if r[0] != "<s>" and r[0] != "</s>"]
result.sort(key=lambda x: x[1])
length = len(result)
return  [session_id] + result[:5]
```

N-Gram

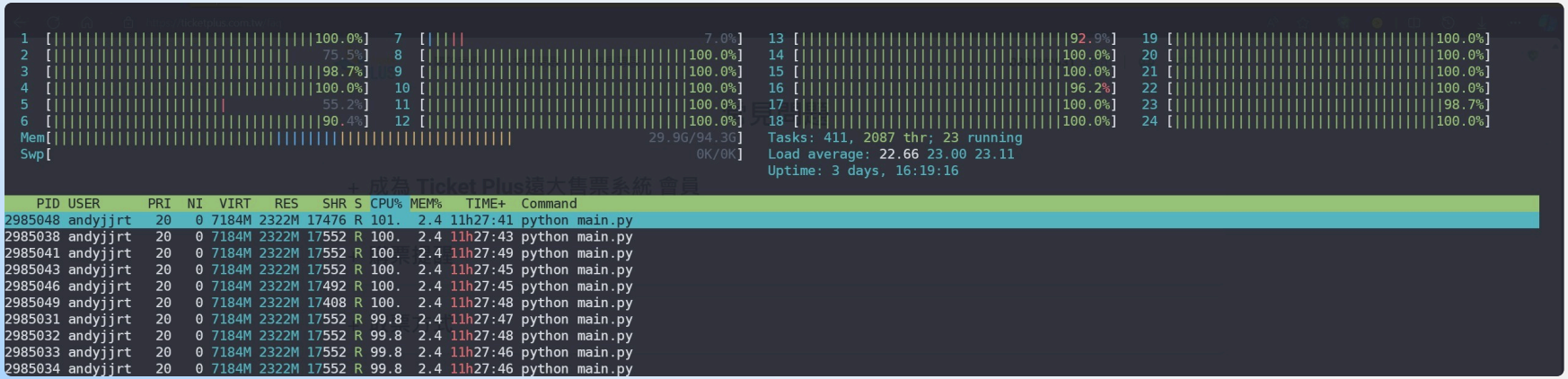
model.score

```
result = list(model.score(words[-2:]))
result = [r for r in result if r[0] != "<s>" and r[0] != "</s>"]
result.sort(key=lambda x: x[1])
length = len(result)
return  [session_id] + result[:5]
```

N-Gram

model.score

```python
result = list(model.score(words[-2:]))
result = [r for r in result if r[0] != "<s>" and r[0] != "</s>"]
result.sort(key=lambda x: x[1])
length = len(result)
return  [session_id] + result[:5]
```

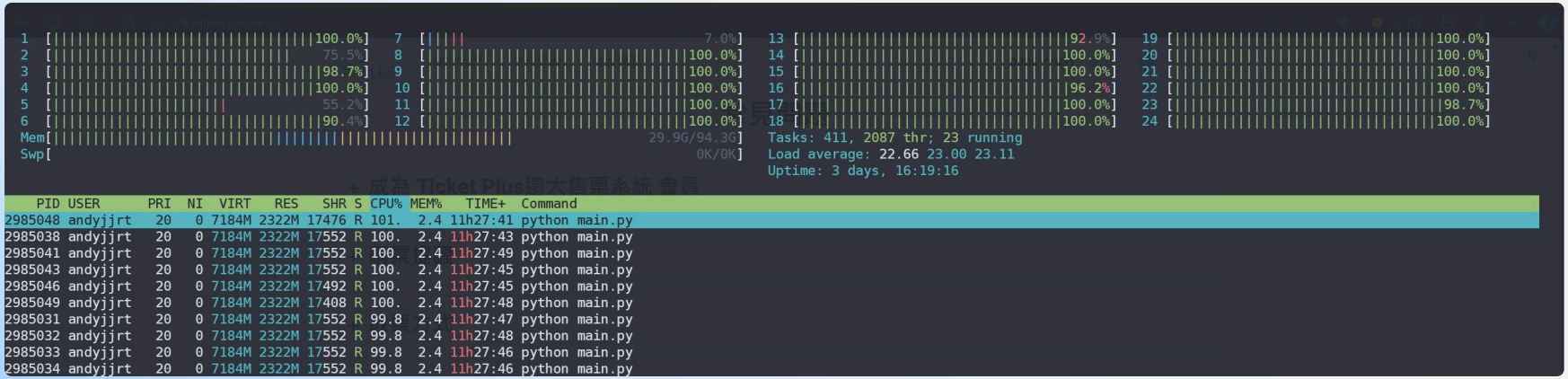


- 12hr per run with 22 threads, **Basic Score: 0.20894**

N-Gram

model.score

```python
result = list(model.score(words[-2:]))
result = [r for r in result if r[0] != "<s>" and r[0] != "</s>"]
result.sort(key=lambda x: x[1])
length = len(result)
return  [session_id] + result[:5]
```



- 12hr per run with 22 threads, **Basic Score: 0.20894**

N-Gram

model.count

```
if words[-1] == words[-2]: return [session_id] + [songIdMapping[words[-1]]] * 5
result = list(model.counts[words[-2:]].items())
result = [r for r in result if r[0] != "<s>" and r[0] != "</s>"]
result.sort(key=lambda x: x[1])
length = len(result)
if length >= 5:
  return  [session_id] + [songIdMapping[r[0]] for r in result[:5]]
else:
  tmp = [songIdMapping[r[0]] for r in result]
  for i in range(5 - len(tmp)):
    tmp.append(random.choice(list(songIdMapping.items()))[1])
  return  [session_id] + tmp
```

N-Gram

model.count

```
if words[-1] == words[-2]: return [session_id] + [songIdMapping[words[-1]]] * 5
result = list(model.counts[words[-2:]].items())
result = [r for r in result if r[0] != "<s>" and r[0] != "</s>"]
result.sort(key=lambda x: x[1])
length = len(result)
if length >= 5:
  return  [session_id] + [songIdMapping[r[0]] for r in result[:5]]
else:
  tmp = [songIdMapping[r[0]] for r in result]
  for i in range(5 - len(tmp)):
    tmp.append(random.choice(list(songIdMapping.items()))[1])
  return  [session_id] + tmp
```

N-Gram

model.count

```
if words[-1] == words[-2]: return [session_id] + [songIdMapping[words[-1]]] * 5
result = list(model.counts[words[-2:]].items())
result = [r for r in result if r[0] != "<s>" and r[0] != "</s>"]
result.sort(key=lambda x: x[1])
length = len(result)
if length >= 5:
  return  [session_id] + [songIdMapping[r[0]] for r in result[:5]]
else:
  tmp = [songIdMapping[r[0]] for r in result]
  for i in range(5 - len(tmp)):
    tmp.append(random.choice(list(songIdMapping.items()))[1])
  return  [session_id] + tmp
```

N-Gram

model.count

```
if words[-1] == words[-2]: return [session_id] + [songIdMapping[words[-1]]] * 5
result = list(model.counts[words[-2:]].items())
result = [r for r in result if r[0] != "<s>" and r[0] != "</s>"]
result.sort(key=lambda x: x[1])
length = len(result)
if length >= 5:
  return  [session_id] + [songIdMapping[r[0]] for r in result[:5]]
else:
  tmp = [songIdMapping[r[0]] for r in result]
  for i in range(5 - len(tmp)):
    tmp.append(random.choice(list(songIdMapping.items()))[1])
  return  [session_id] + tmp
```

N-Gram

model.count

```
if words[-1] == words[-2]: return [session_id] + [songIdMapping[words[-1]]] * 5
result = list(model.counts[words[-2:]].items())
result = [r for r in result if r[0] != "<s>" and r[0] != "</s>"]
result.sort(key=lambda x: x[1])
length = len(result)
if length >= 5:
  return  [session_id] + [songIdMapping[r[0]] for r in result[:5]]
else:
  tmp = [songIdMapping[r[0]] for r in result]
  for i in range(5 - len(tmp)):
    tmp.append(random.choice(list(songIdMapping.items()))[1])
  return  [session_id] + tmp
```

N-Gram

model.count

```
if words[-1] == words[-2]: return [session_id] + [songIdMapping[words[-1]]] * 5
result = list(model.counts[words[-2:]].items())
result = [r for r in result if r[0] != "<s>" and r[0] != "</s>"]
result.sort(key=lambda x: x[1])
length = len(result)
if length >= 5:
  return  [session_id] + [songIdMapping[r[0]] for r in result[:5]]
else:
  tmp = [songIdMapping[r[0]] for r in result]
  for i in range(5 - len(tmp)):
    tmp.append(random.choice(list(songIdMapping.items()))[1])
  return  [session_id] + tmp
```

- 6hr per run with 22 threads, 30% replacement to **0.22006**

N-Gram

What we should do

N-Gram

What we should do

- Randomly choose songs from same `genre` , `titletext` , etc.

N-Gram

What we should do

- Randomly choose songs from same `genre` , `titletext` , etc.
- Use `model.generate()` instead.

N-Gram

What we should do

- Randomly choose songs from same `genre` , `titletext` , etc.
- Use `model.generate()` instead.
- Try 4-gram.

N-Gram

What we should do

- Randomly choose songs from same `genre` , `titletext` , etc.
- Use `model.generate()` instead.
- Try 4-gram.
- Reciprocal Rank Fusion

$$RRFScore(d \in D) = \sum_{r \in R} \frac{1}{k + r(d)}, k = 60$$

N-Gram

What we should do

- Randomly choose songs from same `genre` , `titletext` , etc.
- Use `model.generate()` instead.
- Try 4-gram.
- Reciprocal Rank Fusion

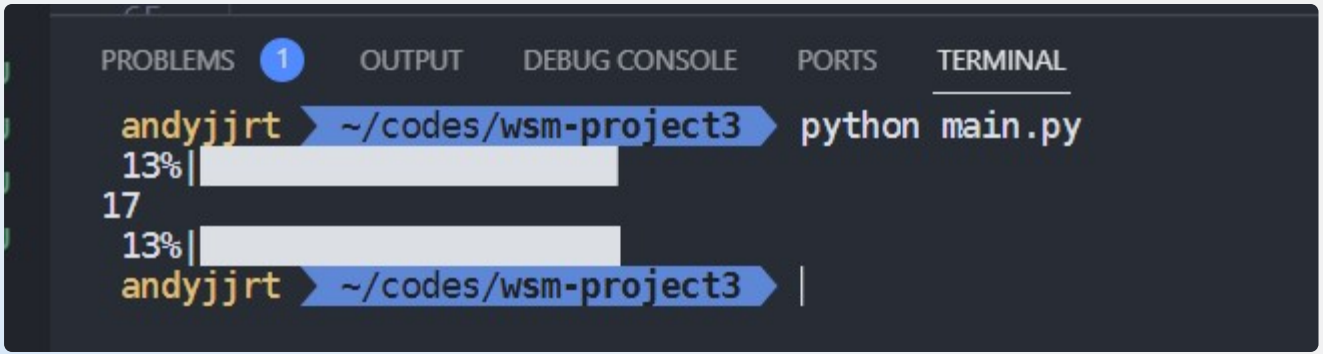
$$RRFScore(d \in D) = \sum_{r \in R} \frac{1}{k + r(d)}, k = 60$$

- Data validation (important).

Discussions & Feedbacks

Discussions & Feedbacks

- Data validation




```
for i in tqdm.tqdm(range(recordLength)):
  words = list()
  for j in range(25):
    words.append(recordTable.loc[i*25 + j, "song_id"])
  allWords.append(words)
```

Thanks for listening