Rohini Vishwanathan
Natural language processing
IST 332

Assignment #3: Feature vectorization and sentiment analysis

Google colab link- https://colab.research.google.com/drive/1CmtYVTmHPOjWtfDQ51GCoJa1Ln4xB1GD?usp=sharing

Section 1 : Cleaned reviews

Lemmatize, drop empties

```
[ ]
def to_lemmas(text):
    if not isinstance(text, str) or not text.strip():
        return ""
    doc = nlp(text.lower())
    toks = [t.lemma_ for t in doc if t.is_alpha and not t.is_stop]
    return " ".join(toks).strip()

df = df_raw.copy()
df["Review_cleaned"] = df["Review"].astype(str).apply(to_lemmas)
df = df[df["Review_cleaned"].str.len() > 0].reset_index(drop=True)

len(df)  # final cleaned row count
```

```
63206
```

Preview (first 5 rows)

```
[5]
✓ 0s
df.loc[:, ["Cust_Rating","Datetime","Review","Restaurant","City","State","Zipcode","Business_Rating_Score","Review
```

	Cust_Rating	Datetime	Review	Restaurant	City	State	Zipcode	Business_Rating_Score	Review_cleaned
0	3.0	2013-12-06 23:22:26	This place is an interesting combo. The chef, ...	La Mongerie Bakery & Bistro	Atlanta	GA	30308	3.0	place interesting combo chef david incredible ...
1	5.0	2008-11-16 09:44:04	Pizza Hut is great! You get huge pizzas for yo...	Pizza Hut	Orlando	FL	32819	3.5	pizza hut great huge pizza money service quick...
2	5.0	2016-04-11 17:00:34	Always a great place to taste some tea with fr...	Teavana	Atlanta	GA	30326	3.0	great place taste tea friend staff friendly pushy
3	4.0	2015-05-26 06:47:56	Delish! Stopped in on a random Sunday and so g...	Native Foods Cafe	Happy Valley	OR	97086	4.5	delish stop random sunday glad normally eat he...
4	2.0	2014-02-23 23:30:56	The 2 stars are for great variety. I would hav...	Waltham India Market	Waltham	MA	02453	3.0	star great variety give star remove failure in...

Section 2: FastText (pre-trained)

First 5 reviews with the new 300d FastText vectors+ original metadata:

Assemble dataframe (metadata + vectors) & preview

```
meta_cols = ["Cust_Rating","Datetime","Review","Restaurant","City","State","Zipcode","Business_Rating_Score"]
vec_cols  = [f"ft_{i}" for i in range(ft_dim)]

df_ft = pd.concat(
    [df[meta_cols].reset_index(drop=True),
     pd.DataFrame(X, columns=vec_cols)],
    axis=1
)

df_ft.head(5)
#df_ft.to_csv("/content/drive/MyDrive/Mydata/df_fasttext_review_vectors.csv", index=False)
```

	Cust_Rating	Datetime	Review	Restaurant	City	State	Zipcode	Business_Rating_Score	ft_0	ft_1	...	ft_290	ft_291	ft_292	ft_293	ft_294	ft_295	ft_296	ft_297	ft_298	ft_299
0	3.0	2013-12-06 23:22:26	This place is an interesting combo. The chef, ...	La Mongerie Bakery & Bistro	Atlanta	GA	30308	3.0	-0.028984	-0.037984	...	0.004486	0.032442	-0.024005	-0.006096	0.002617	-0.008675	0.006830	0.022984	-0.027191	-0.006683
1	5.0	2008-11-16 09:44:04	Pizza Hut is great! You get huge pizzas for yo...	Pizza Hut	Orlando	FL	32819	3.5	-0.023470	-0.014872	...	0.015916	0.024685	-0.003450	-0.015848	0.021220	-0.019094	0.032824	0.034016	-0.009846	-0.013066
2	5.0	2016-04-11 17:00:34	Always a great place to taste some tea with fr...	Teavana	Atlanta	GA	30326	3.0	-0.016991	-0.042579	...	-0.021573	0.058634	-0.019549	0.008980	0.000755	0.003944	0.033365	0.042965	-0.018749	0.006130
3	4.0	2015-05-26 06:47:56	Delish! Stopped in on a random Sunday and so g...	Native Foods Cafe	Happy Valley	OR	97086	4.5	-0.012254	0.006191	...	0.008091	0.011410	-0.026101	-0.010715	0.013988	-0.021434	0.023488	0.008143	-0.019713	-0.001002
4	2.0	2014-02-23 23:30:56	The 2 stars are for great variety. I would hav...	Waltham India Market	Waltham	MA	02453	3.0	0.000793	-0.003037	...	-0.004901	0.031194	-0.016404	0.002264	-0.001636	-0.024458	0.014530	0.005403	-0.013201	-0.006987

5 rows × 308 columns

Section 3 : Custom Word2Vec + Comparison

Top 20 unigrams and top 20 bigrams (with counts)

Top 20 unigrams and top 20 bigrams

```
from collections import Counter

uni_counter = Counter([w for doc in tokens for w in doc])
bi_counter  = Counter([w for doc in tokens_bi for w in doc if "_" in w])

top20_uni = uni_counter.most_common(20)
top20_bi  = bi_counter.most_common(20)

top20_uni, top20_bi
```

```
([('food', 46477),
  ('good', 44839),
  ('place', 37341),
  ('order', 35217),
  ('time', 24904),
  ('like', 24834),
  ('great', 24331),
  ('come', 22600),
  ('service', 21267),
  ('try', 17593),
  ('get', 17105),
  ('go', 16434),
  ('chicken', 15061),
  ('restaurant', 14686),
  ('eat', 13570),
  ('pizza', 12481),
  ('love', 12292),
  ('nice', 11317),
  ('wait', 10768),
  ('want', 10567)],
 [('customer_service', 2554),
  ('feel_like', 1844),
  ('highly_recommend', 1761),
  ('ice_cream', 1522),
  ('staff_friendly', 1409),
  ('wait_minute', 1162),
  ('happy_hour', 1158),
  ('look_forward', 826),
  ('friendly_staff', 818),
  ('super_friendly', 772),
  ('take_minute', 721),
  ('gluten_free', 693),
  ('parking_lot', 691),
  ('dim_sum', 671),
  ('decide_try', 646),
  ('little_bit', 614),
  ('mac_cheese', 602),
  ('price_reasonable', 592),
  ('minute_later', 586),
  ('year_ago', 575)])
```

Train Word2Vec:

Train Word2Vec (CBOW, min_count=30, window=7, size=300, workers=2, epochs=20)

```
from gensim.models import Word2Vec

w2v = Word2Vec(
    sentences=tokens_bi,
    vector_size=300,
    window=7,
    min_count=30,
    workers=2,
    sg=0,               # CBOW
    epochs=20
)

# keyed vectors handle
w2v_kv = w2v.wv
w2v_kv.vector_size
```

```
300
```

Most similar words (10) for the top two frequent words, from BOTH models:

Most-similar examples (unigram + bigram)

```
def try_most_similar(model, word, topn=10):
    if word in model.key_to_index:
        return model.most_similar(word, topn=topn)
    else:
        return f"{word} not in vocabulary"

examples = {
    "food":  try_most_similar(w2v_kv, "food", 10),
    "service": try_most_similar(w2v_kv, "service", 10),
    # example bigram token (may vary by corpus; safe-check)
    "great_service": try_most_similar(w2v_kv, "great_service", 10),
}
examples
```

```
{'food': [('service', 0.415457546710968),
  ('restaurant', 0.34035512804985046),
  ('meal', 0.33094701170921326),
  ('speedy', 0.3276204764842987),
  ('extremely_slow', 0.3148893713951111),
  ('sushi', 0.31467074155807495),
  ('server_attentive', 0.306128591299057),
  ('chip_salsa', 0.30594488978385925),
  ('time', 0.3056448698043823),
  ('place', 0.30493712425231934)],
 'service': [('customer_service', 0.5901476740837097),
  ('costumer_service', 0.5624114274978638),
  ('waitstaff', 0.5535829067230225),
  ('staff', 0.4810692071914673),
  ('service_prompt', 0.4769642651081085),
  ('server_attentive', 0.47683045268058777),
  ('food', 0.4154576063156128),
  ('experience', 0.40347057580947876),
  ('refill_drink', 0.39816126227378845),
  ('server', 0.39810675382614136)],
 'great_service': 'great_service not in vocabulary'}
```

Compare similarities: FastText vs Custom Word2Vec:

```
import numpy as np
import pandas as pd

def cos_sim(a, b):
    denom = (np.linalg.norm(a) * np.linalg.norm(b))
    if denom == 0: return np.nan
    return float(np.dot(a, b) / denom)

# pick pairs from frequent terms so they're likely in both
pairs = [
    ("food", "service"),
    ("food", "price"),
    ("good", "bad"),
    ("wait", "time"),
]

rows = []
for a, b in pairs:
    # skip if OOV in either model
    if (a in w2v_kv.key_to_index) and (b in w2v_kv.key_to_index) and \
       (a in ft.key_to_index) and (b in ft.key_to_index):
        rows.append({
            "term_a": a,
            "term_b": b,
            "sim_w2v": cos_sim(w2v_kv.get_vector(a), w2v_kv.get_vector(b)),
            "sim_fasttext": cos_sim(ft.get_vector(a), ft.get_vector(b)),
        })

sim_df = pd.DataFrame(rows)
sim_df
```

	term_a	term_b	sim_w2v	sim_fasttext
0	food	service	0.415458	0.483610
1	food	price	0.236316	0.467126
2	good	bad	0.319150	0.850309
3	wait	time	0.460444	0.552057

FastText nearest neighbors (same seeds)

```
[==================================================] 100.0% 958.5/958.4MB downloaded
{'food (FT)': [('foods', 0.7766500115394592),
  ('food-', 0.7599727511405945),
  ('foodstuff', 0.735816240310669),
  ('foodless', 0.7320124506950378),
  ('food.', 0.7302034497261047),
  ('foodstuffs', 0.7282736897468567),
  ('healthfood', 0.7277587652206421),
  ('dogfood', 0.7183505892753601),
  ('catfood', 0.716797411441803),
  ('non-food', 0.7167892456054688)],
 'service (FT)': [('services', 0.7911463975906372),
  ('non-service', 0.7496036291122437),
  ('service-', 0.7314968109130859),
  ('servic', 0.7274243235588074),
  ('service-type', 0.7205984592437744),
  ('post-service', 0.7153797745704651),
  ('sevice', 0.7121303081512451),
  ('service--and', 0.7114466428756714),
  ('cross-service', 0.7097828984260559),
  ('service-wide', 0.6980454325675964)],
 'customer_service (FT)': 'customer_service not in FastText vocab'}
```

Comparison:
FastText retrieves broader, morphology-heavy neighbors (e.g., plural/variant forms and general terms), while the custom Word2Vec surfaces domain-specific co-occurrences tied to restaurant reviews (e.g., staff/service phrases and common collocations). In our cosine table, FastText tends to assign higher similarity to broadly related pairs (e.g., *good–bad*), whereas the custom model keeps stronger separation that better reflects sentiment polarity in this corpus. Net: FastText offers robust coverage; the custom model captures dataset-specific usage that's often more useful for downstream sentiment/restaurant tasks.

Section 4: Per-restaurant document embeddings (1075 rows)
First 5 rows of the business-level embeddings data frame

```python
import pandas as pd
import numpy as np

# vector columns come from df_ft created in Section 2
vec_cols = [c for c in df_ft.columns if c.startswith("ft_")]
meta_cols = ["Restaurant","City","State","Zipcode","Business_Rating_Score","Cust_Rating","Review"]

df_with_vec = pd.concat(
    [df[meta_cols].reset_index(drop=True), df_ft[vec_cols].reset_index(drop=True)],
    axis=1
)

group_cols = ["Restaurant","City","State","Zipcode","Business_Rating_Score"]
agg_map = {c: "mean" for c in vec_cols}
agg_map.update({"Cust_Rating": "mean", "Review": "count"})

df_rest_doc = (
    df_with_vec
      .groupby(group_cols, as_index=False)
      .agg(agg_map)
      .rename(columns={"Review":"review_count","Cust_Rating":"cust_rating_avg"})
      .sort_values(["State","City","Restaurant"])
      .reset_index(drop=True)
)

len(df_rest_doc), df_rest_doc.shape
```

```
(1075, (1075, 307))
```

Preview first 5 restaurants

[18] ✓ 0s

```python
df_rest_doc.loc[:, ["Restaurant","City","State","Zipcode","Business_Rating_Score",
                    "review_count","cust_rating_avg"] + vec_cols[:5]].head(5)
```

	Restaurant	City	State	Zipcode	Business_Rating_Score	review_count	cust_rating_avg	ft_0	ft_1	ft_2	ft_3	ft_4
0	IHOP	BURNABY	BC	V5H 2E6	3.5	60	3.566667	-0.006099	-0.020242	0.013036	-0.000374	-0.010482
1	Cattle Cafe	Burnaby	BC	V5H 4T2	2.5	73	2.602740	-0.011275	-0.017237	0.007729	-0.002314	-0.006924
2	Chatime	Burnaby	BC	V5H 2E8	3.0	98	2.948980	-0.006267	-0.025750	0.003499	-0.002189	-0.003196
3	Chronic Tacos	Burnaby	BC	V5H 4P1	3.0	32	3.031250	-0.017612	-0.010860	0.001035	0.003337	-0.016227
4	Cozmos Cafe + Bistro	Burnaby	BC	V5B 1S1	4.0	81	4.148148	-0.013299	-0.021173	0.013922	-0.000054	-0.014217

Section 5: Sentiment analysis per restaurant (1075 rows)

Aggregate by restaurant (⟶ 1,075 rows) + preview

```
group_cols = ["Restaurant","City","State","Zipcode","Business_Rating_Score"]

df_rest_sa = (
    df.groupby(group_cols, as_index=False)
      .agg(
          review_count=("Review","count"),
          polarity=("tb_polarity","mean"),
          subjectivity=("tb_subjectivity","mean"),
          nltk_compound=("nltk_compound","mean"),
          cust_rating_avg=("Cust_Rating","mean"),
      )
      .sort_values(["State","City","Restaurant"])
      .reset_index(drop=True)
)

len(df_rest_sa), df_rest_sa.shape, df_rest_sa.head(5)
```

```
(1075,
 (1075, 10),
              Restaurant     City State  Zipcode  Business_Rating_Score  \
0                  IHOP  BURNABY    BC  V5H 2E6                    3.5
1           Cattle Cafe  Burnaby    BC  V5H 4T2                    2.5
2              Chatime  Burnaby    BC  V5H 2E8                    3.0
3         Chronic Tacos  Burnaby    BC  V5H 4P1                    3.0
4  Cozmos Cafe + Bistro  Burnaby    BC  V5B 1S1                    4.0

   review_count  polarity  subjectivity  nltk_compound  cust_rating_avg
0            60  0.182035      0.576365       0.685772         3.566667
1            73  0.041021      0.588100       0.360289         2.602740
2            98  0.110853      0.563575       0.572917         2.948980
3            32  0.130774      0.574958       0.573644         3.031250
4            81  0.351373      0.618779       0.872700         4.148148  )
```

Per-review sentiment (VADER + TextBlob)

```
# install once per fresh runtime
import sys, subprocess
subprocess.check_call([sys.executable, "-m", "pip", "install", "-q", "vaderSentiment==3.3.2", "textblob==0.18.0"])

from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
from textblob import TextBlob
analyzer = SentimentIntensityAnalyzer()

# compound ∈ [-1,1]; TextBlob polarity ∈ [-1,1]; subjectivity ∈ [0,1]
df["nltk_compound"]  = df["Review_cleaned"].astype(str).map(lambda t: analyzer.polarity_scores(t)["compound"])
df["tb_polarity"]    = df["Review_cleaned"].astype(str).map(lambda t: TextBlob(t).sentiment.polarity)
df["tb_subjectivity"]= df["Review_cleaned"].astype(str).map(lambda t: TextBlob(t).sentiment.subjectivity)

df[["Review","nltk_compound","tb_polarity","tb_subjectivity"]].head(3)
```

	Review	nltk_compound	tb_polarity	tb_subjectivity
0	This place is an interesting combo. The chef, ...	0.9506	0.369913	0.499026
1	Pizza Hut is great! You get huge pizzas for yo...	0.9623	0.555208	0.725000
2	Always a great place to taste some tea with fr...	0.8555	0.587500	0.625000