A Modern Recommender System using Aspect Based Sentiment Analysis

Kevin Joseph Scaria

Department of Information Technology, Arizona State University

IFT 598: Natural Language Processing

Dr. Brian Atkinson

December 04, 2022

Table of Contents

It’s the Final Countdown

1. Introduction

The world has been rapidly evolving since the advent of the Internet and the content available on the web has increased exponentially. In recent times, ever since E-commerce came into existence, consumers who interact with E-commerce platforms have the need to be recommended products/content personalized to their interaction history. Recommender systems are an important part of any consumer-facing system as it benefits both the consumer as well as the firms.

In this project, we propose a modern recommender system based on textual reviews available on the E-commerce platforms such as Amazon. Traditional recommender systems often recommend relevant items based on consumer interaction and their similarity with other users on the platform. But they disregard rich information available in the form of reviews. These reviews have information about how similar users have felt about a product/content and their implicit/qualitative factors associated with the product/content which can be mined using Aspect Based Sentiment Analysis (ABSA). Thus, as part of the project, we develop a system that can answer the **questions** such as *“What are the top laptops that have a good display, screen-resolution?*” or *“What are the top laptops that are good for gaming?*”. Such a system helps in recommending products based on other users who have tried the product from the platform and have experienced the features of the product.

2. Literature Review

Wanxiang Che et al., 2015 suggested that syntactic characteristics play a large role in aspect-based sentiment analysis. The reviews that this assignment focuses on, nevertheless, are organic and unplanned, which makes them difficult for syntactic parsers to understand. They provide a paradigm in which the aspect-based sentiment analysis is preceded by the addition of a sentiment sentence compression (Sent Comp) phase. To automatically compress emotion sentences, they also proposed using a discriminative conditional random field model with a few unique features. Their approach considerably enhanced the performance of aspect-based sentiment analysis using the Chinese corpora of four product areas.

Wenya Wang et al., 2016 proposed a novel approach to utilize associations between aspect and opinion phrases. The suggested model simultaneously learns high-level discriminative features and double propagates data between terms expressing an opinion and terms expressing an aspect. The proposed method is also flexible to integrate custom characteristics.

Md ShadAkhtar et al., 2017 proposed an ensemble-based model with cascading. They employed features that have been discovered in light of the characteristics of various classifiers and domains. Maximum Entropy (ME), Conditional Random Field (CRF), and Support Vector Machines were the primary learning algorithms. The efficacy of their suggested strategy was demonstrated in two different types of domains.

Choe Yang et al., 2019, asserted that the goal of aspect-based sentiment analysis is to identify the polarities of certain targets' sentiments inside a given text and it is unfair to compute the attention score for context using an average vector of the target. They suggested a Coattention-LSTM network that simultaneously learns nonlinear context and target representations.

Kai Wang et al., 2020 suggested that using attention-based neural network models frequently conflates the linkages because of language complexity and the presence of several features in a single sentence. They proposed using efficient syntactic information encoding to solve this issue using a pruned standard dependency parse tree to build a unified aspect-oriented dependency tree structure rooted at a target aspect. This new encoded tree structure is used for sentiment prediction and coined as a relational graph attention network (R-GAT).

Mei Zhen Liu et al., 2021 proposed an interactive co-attention mechanism to capture the interactions between aspect and context, which interactively concentrates the semantic influences on context and aspect to generate a more informative representation.

3. Tools

The tools that we will be using are described below:

Python: The coding language for the project.

I. Nltk: Python library which may be used for stop-word removal and other text pre-processing tasks such as lemmatization and tokenization. This library has functions to perform pre-processing and the output is the preprocessed text after each step. The pros of this tool are that it is extremely fast, however, since the process is rule-based it often pre-processes the task incorrectly.

II. TextBlob, VADER: Python libraries that will be used for the aspect sentiment classification tasks. The tool is extremely fast and also computes a sentiment score for sentiment analysis tasks. This tool provides sentiment scores based on a rule-based limited corpus.

III. Huggingface: We use large language models (LLM) for word embeddings and thus we might use the hugging face hub which has downloadable weights for LLMs such as BERT, T5, etc. The advantage of this library is easy to access pre-trained weights from large language models. The drawback is difficulty in customizing the model fine-tuning and architecture of models.

IV. Google Colab or Jupyter notebook: We will be using colab notebooks which are Jupyter notebooks provided by Google. The advantage is we can use free GPU computing resources for finetuning large language models. Cons is there is a run-time limit.

4. Data

The datasets that are used for this project include (1) SemEval 2014 Task 4: Aspect-based sentiment analysis (Potinki et al., 2014), (2) Target Oriented Opinion Word Extraction (Fan et al., 2019), and (3) Amazon reviews data (Ni et al., 2019). These are standard datasets proposed to further research in the field of aspect-based sentiment analysis and recommender systems.

Dataset (1) provided has 6086 training reviews (3041 from the restaurants' domain, 3045 from the laptops domain) and 1600 test reviews (800 from the restaurant's domain, 800 from the laptops domain). The data has been made publicly available at the URL. The data provided is in XML format and would require parsing the XML tree to bring it into a usable format. The data is reviewed from several users about restaurants and laptops. Since the name and other details of these users are not provided, there is no sensitive information present in the data and thus there is no need to consider steps to ensure data privacy. This dataset was used to train the Aspect Term Extraction (ATE) model.

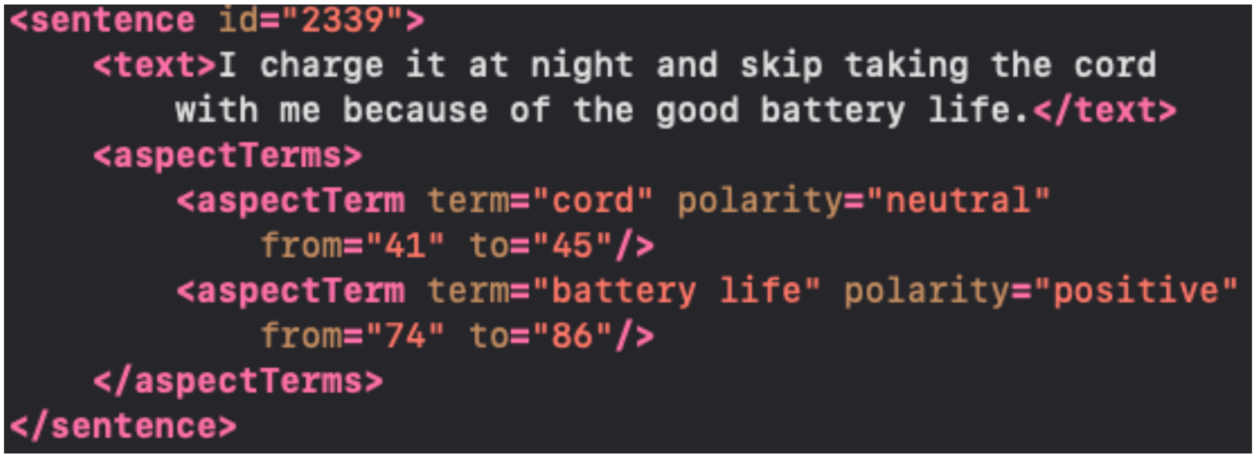
Dataset (2) has a small subset of data extracted from Dataset (1) and labeled by the authors of the paper "Target-oriented opinion words extraction with Target-fused neural sequence labeling" (Fan et al., 2019). This has around 1.5k samples for the laptops domains which were utilized to train the Target Oriented Opinion Words Extraction (TOWE) model. This data is publicly available at the URL.

Dataset (3) is a very large corpus of amazon reviews. There are two tables viz. reviews and metadata. The review data is a sequence of reviews for various products. The metadata table has information regarding the product, for example, the name and price of the product. We use a subset of the data specifically for Electronics. The paper proposed the dataset as part of "Justifying recommendations using distantly-labeled reviews and fine-grained aspects" (Ni et al., 2019). Further specifics about this dataset are mentioned in the data cleaning section. This data is publicly available at the URL.

5. Data Cleaning

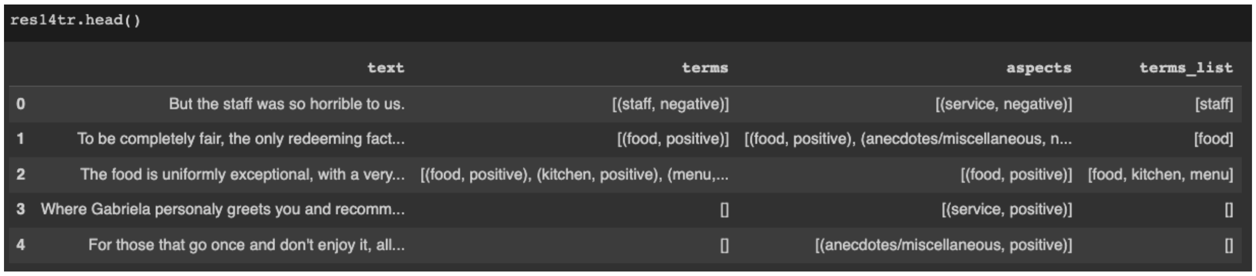
5.1. Dataset 1 – ATE data

Dataset (1) was in XML format as shown below which had to be parsed using python and brought into the required tabular format.



```
<sentence id="2339">
    <text>I charge it at night and skip taking the cord
        with me because of the good battery life.</text>
    <aspectTerms>
        <aspectTerm term="cord" polarity="neutral"
            from="41" to="45"/>
        <aspectTerm term="battery life" polarity="positive"
            from="74" to="86"/>
    </aspectTerms>
</sentence>
```

This was later transformed to a tabular data format as shown below:



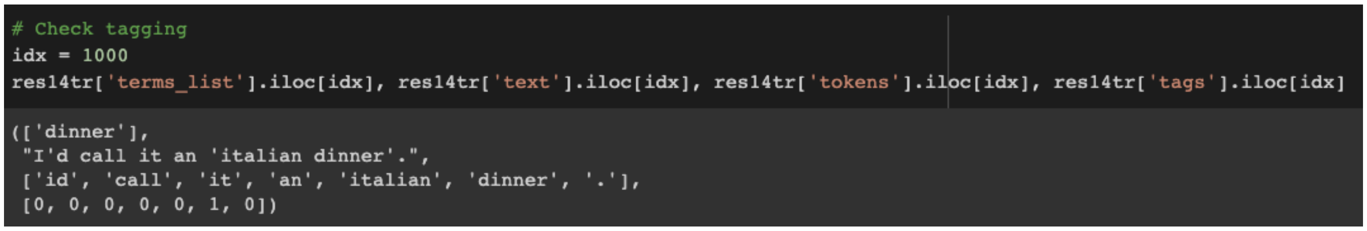
```
res14tr.head()
```

	text	terms	aspects	terms_list
0	But the staff was so horrible to us.	[(staff, negative)]	[(service, negative)]	[staff]
1	To be completely fair, the only redeeming fact...	[(food, positive)]	[(food, positive), (anecdotes/miscellaneous, n...	[food]
2	The food is uniformly exceptional, with a very...	[(food, positive), (kitchen, positive), (menu,...	[(food, positive)]	[food, kitchen, menu]
3	Where Gabriela personaly greets you and recomm...	[]	[(service, positive)]	[]
4	For those that go once and don't enjoy it, all...	[]	[(anecdotes/miscellaneous, positive)]	[]

Since the dataset (1) was used for the Aspect Term Extraction (ATE) subtask, we had to convert this tabular format and create token tags in the form of [0] and [1] for each token, where a [1] would represent that the token is an aspect term. The token tags were created by tokenizing the sentence and marking all token ids as 0 which did not have the corresponding aspect term. The code for creating tags in the required format is shown below.

```
def add_tags(dataframe):
    tokens_list, tags_list = [], []
    for idx, asp_terms in
tqdm(enumerate(dataframe['terms_list'].iloc[:])):
        sent = dataframe['text'].iloc[idx].replace("'", "")
        sent_tokens = word_tokenize(sent.lower())
        tags = [0]*len(sent_tokens)
        for asp_term in asp_terms:
            if asp_term.lower() in sent_tokens:
                req_idx = sent_tokens.index(asp_term.lower())
                tags[req_idx] = 1
        tokens_list.append(sent_tokens)
        tags_list.append(tags)

    dataframe['tokens'] = tokens_list
    dataframe['tags'] = tags_list
    return dataframe
```

An example of sample review 1000 is shown below. Here the token *"dinner"* is an aspect term thus it has a label 1 and all other tokens have a 0 label.



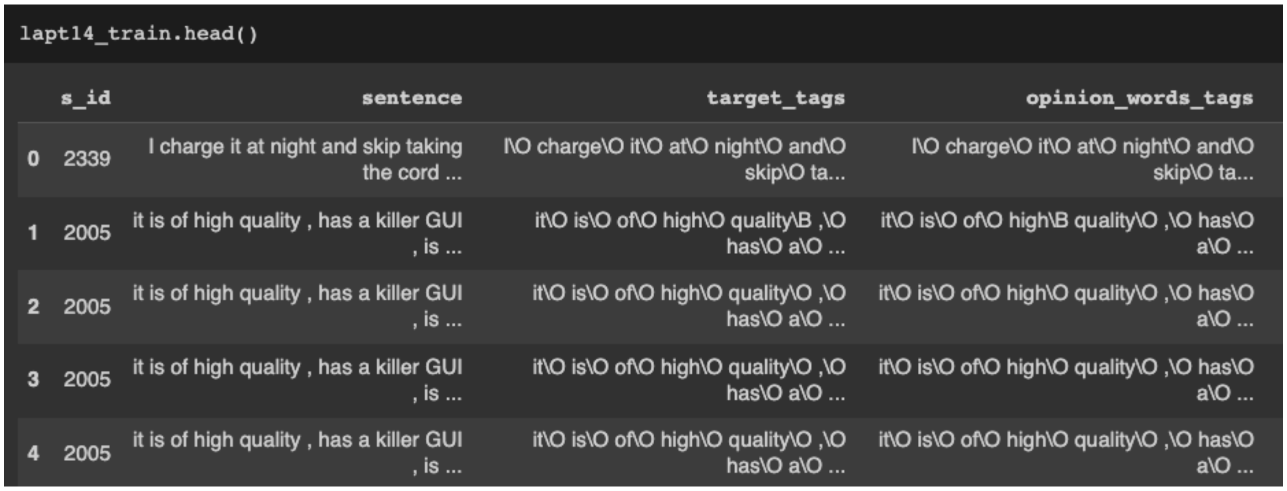
```
# Check tagging
idx = 1000
res14tr['terms_list'].iloc[idx], res14tr['text'].iloc[idx], res14tr['tokens'].iloc[idx], res14tr['tags'].iloc[idx]
```

```
(['dinner'],
 "I'd call it an 'italian dinner'.",
 ['id', 'call', 'it', 'an', 'italian', 'dinner', '.'],
 [0, 0, 0, 0, 0, 1, 0])
```

With this step, the final data creation for ATE task is completed.

5.2. Dataset 2 – TOWE data

The dataset (2) which was used for TOWE already had token tags which signified if a token is an opinion word or not. However, the format was different from what is required by the Huggingface BERT token classification problem. Specifically, the issue was in the raw data the token tags were provided as "It/O was/O a/O good/B laptop/O". Here the /O and /B tags indicate not an opinion word and an opinion word respectively. However hugging face expects the tags to be a list [O, O, O, B, O] which means the same thing. The initial format is shown below.



```
lapt14_train.head()
```

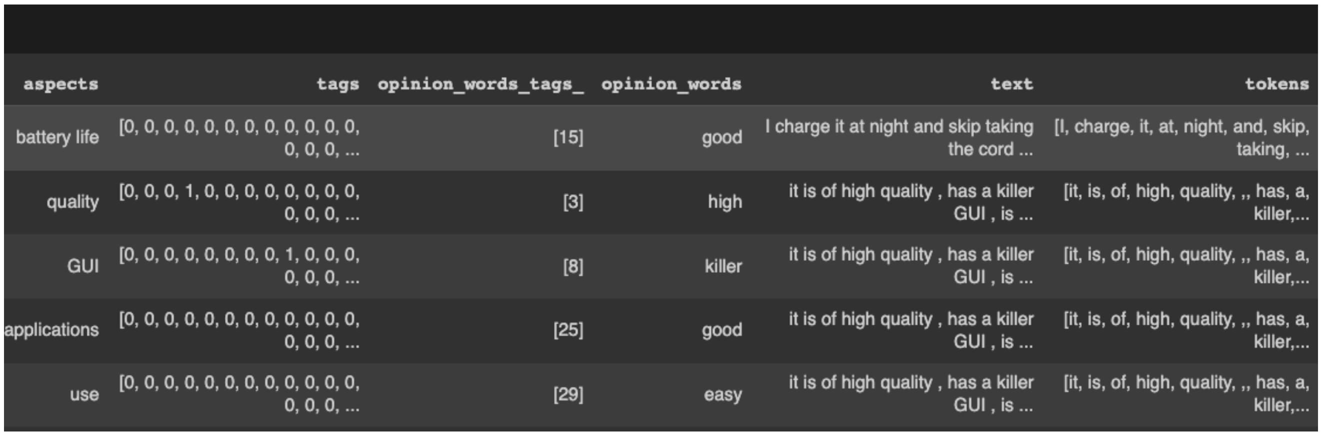
	s_id	sentence	target_tags	opinion_words_tags
0	2339	I charge it at night and skip taking the cord ...	I\O charge\O it\O at\O night\O and\O skip\O ta...	I\O charge\O it\O at\O night\O and\O skip\O ta...
1	2005	it is of high quality , has a killer GUI , is ...	it\O is\O of\O high\O quality\B ,\O has\O a\O ...	it\O is\O of\O high\B quality\O ,\O has\O a\O ...
2	2005	it is of high quality , has a killer GUI , is ...	it\O is\O of\O high\O quality\O ,\O has\O a\O ...	it\O is\O of\O high\O quality\O ,\O has\O a\O ...
3	2005	it is of high quality , has a killer GUI , is ...	it\O is\O of\O high\O quality\O ,\O has\O a\O ...	it\O is\O of\O high\O quality\O ,\O has\O a\O ...
4	2005	it is of high quality , has a killer GUI , is ...	it\O is\O of\O high\O quality\O ,\O has\O a\O ...	it\O is\O of\O high\O quality\O ,\O has\O a\O ...

The code to transform the tags to the required format is shown below. The preprocess function also handles the aspect term (target) infusion into the review text so that the language models understand the context with respect to the target term. This is done by appending a prompt to every review text: *review_text $_1$+ "The aspect term is ${aspect_term1}"*. In doing this, the large language models learn contextual representations to sequence tag the opinion word for the corresponding aspect term.

```
def preprocess(dataframe):
    dataframe['sentence_tokens'] = dataframe['sentence'].apply(lambda x:
word_tokenize(x))

    dataframe['target_tags_'] = dataframe['target_tags'].apply(lambda x:
[i.split('\\')[-1] for i in x.split()])
```

```
    dataframe['target_tags_'] = dataframe['target_tags_'].apply(lambda x:
[idx for idx, i in enumerate(x) if i in ['B', 'I']])
    dataframe['aspects'] = dataframe[['sentence_tokens',
'target_tags_']].apply(lambda x: ' '.join([x[0][i] for i in x[1]]),
axis=1)

    opinion_label_map = {'O':0, 'B':1, 'I':1}

    dataframe['tags'] = dataframe['opinion_words_tags'].apply(lambda x:
[opinion_label_map[i.split('\\')[-1]] for i in x.split()])
    dataframe['opinion_words_tags_'] = dataframe['tags'].apply(lambda x:
[idx for idx, i in enumerate(x) if i in [1]])
    dataframe['opinion_words'] = dataframe[['sentence_tokens',
'opinion_words_tags_']].apply(lambda x: ' '.join([x[0][i] for i in x[1]]),
axis=1)

    # Encode the target aspect word into the sentence
    dataframe['sentence_token_len'] = dataframe['sentence'].apply(lambda
x: len(word_tokenize(x)))
    dataframe['text'] = dataframe[['sentence', 'aspects']].apply(lambda x:
x[0] + f" The aspect identified is: {x[1]}", axis=1)
    dataframe['tokens'] = dataframe['text'].apply(lambda x:
word_tokenize(x))
    dataframe['text_token_len'] = dataframe['tokens'].apply(lambda x:
len(x))

    # Add additional 0 labels to opinion_words_tags_label to match the
length of input text tokens
    dataframe['additional_tags_len'] = dataframe['text_token_len'] -
dataframe['sentence_token_len']
    dataframe['tags'] = dataframe[['tags',
'additional_tags_len']].apply(lambda x: x[0] + [0]*x[1], axis=1)
    dataframe.drop(['additional_tags_len', 'text_token_len',
'sentence_token_len'], axis=1, inplace=True)

    # Removing files with incorrect tags and tokens length mismatch
    dataframe = dataframe[dataframe[['tags', 'tokens']].apply(lambda x:
len(x[0])==len(x[1]), axis=1)]
    return dataframe
```

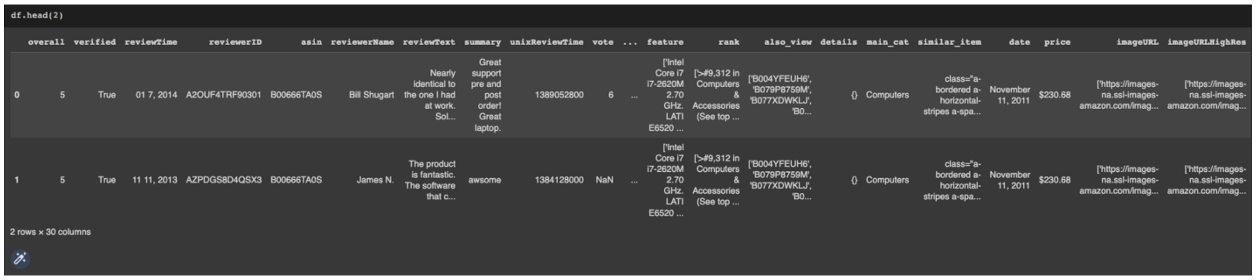
The final transformed data for the TOWE task is shown below:



aspects	tags	opinion_words_tags_	opinion_words	text	tokens
battery life	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[15]	good	I charge it at night and skip taking the cord ...	[I, charge, it, at, night, and, skip, taking, ...
quality	[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[3]	high	it is of high quality , has a killer GUI , is ...	[it, is, of, high, quality, ,, has, a, killer,...
GUI	[0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, ...	[8]	killer	it is of high quality , has a killer GUI , is ...	[it, is, of, high, quality, ,, has, a, killer,...
applications	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[25]	good	it is of high quality , has a killer GUI , is ...	[it, is, of, high, quality, ,, has, a, killer,...
use	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[29]	easy	it is of high quality , has a killer GUI , is ...	[it, is, of, high, quality, ,, has, a, killer,...

5.3. Dataset 3 – Amazon reviews data

This dataset is available in JSON format and is very large to load into memory. Just the electronics section alone was 18GB. This had the metadata as well as review data as two separate objects. Initially, the metadata was loaded in chunks, and all the products that were in the laptops section were filtered. Similarly, the review data was loaded in chunks. Each chunk was iteratively INNER joined with the final product data to extract reviews related to the laptops that were earlier mined from the metadata object. Once all the reviews were extracted for all the filtered products, it was saved as a CSV file. The final data frame looks as shown below:



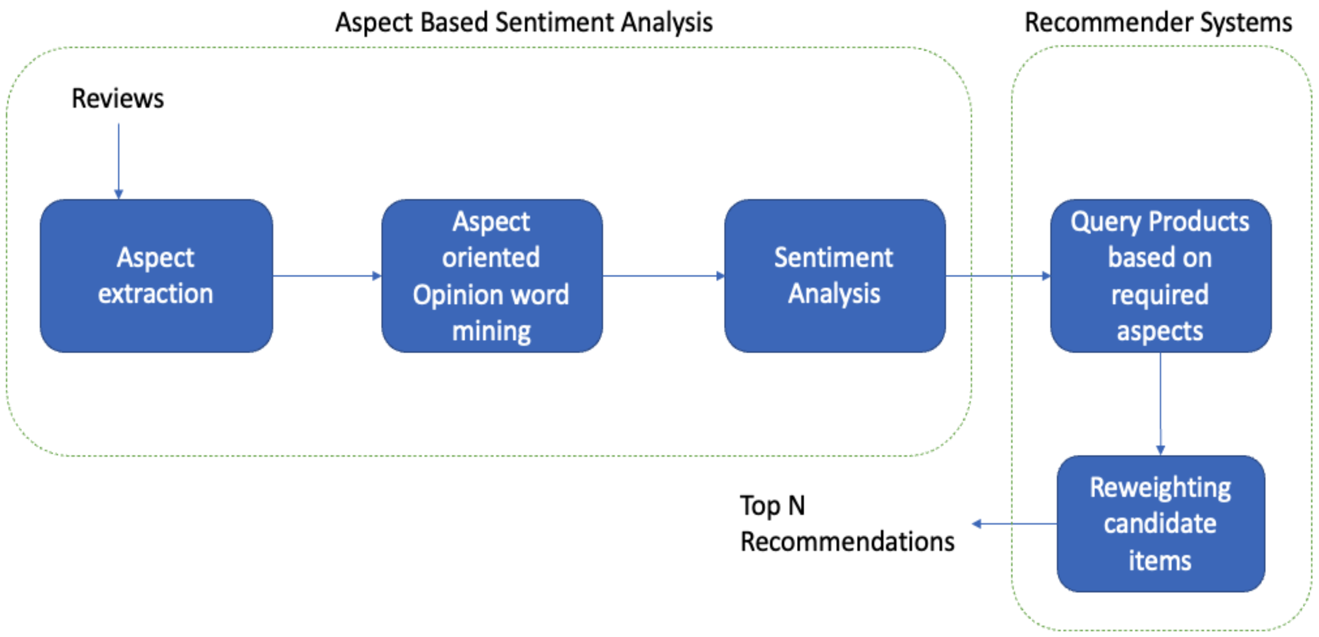
```
df.head(2)
```

	overall	verified	reviewTime	reviewerID	asin	reviewerName	reviewText	summary	unixReviewTime	vote	...	feature	rank	also_view	details	main_cat	similar_item	date	price	imageURL	imageURLHighRes
0	5	True	01 7, 2014	A2OUF4TRF90301	B00666TA0S	Bill Shugart	Nearly identical to the one I had at work. Sol...	Great support pre and post order! Great laptop.	1389052800	6	...	['Intel Core i7 i7-2620M 2.70 GHz. LATI E6520 ...	['>#9,312 in Computers & Accessories (See top ...	['B004YFEUH6', 'B079P8759M', 'B077XDWKLJ', 'B0...	{}	Computers	class="a-bordered a-horizontal-stripes a-spa...	November 11, 2011	$230.68	['https://images-na.ssl-images-amazon.com/imag...	['https://images-na.ssl-images-amazon.com/imag...
1	5	True	11 11, 2013	AZPDGS8D4QSX3	B00666TA0S	James N.	The product is fantastic. The software that c...	awsome	1384128000	NaN	...	['Intel Core i7 i7-2620M 2.70 GHz. LATI E6520 ...	['>#9,312 in Computers & Accessories (See top ...	['B004YFEUH6', 'B079P8759M', 'B077XDWKLJ', 'B0...	{}	Computers	class="a-bordered a-horizontal-stripes a-spa...	November 11, 2011	$230.68	['https://images-na.ssl-images-amazon.com/imag...	['https://images-na.ssl-images-amazon.com/imag...

2 rows × 30 columns

6. Process

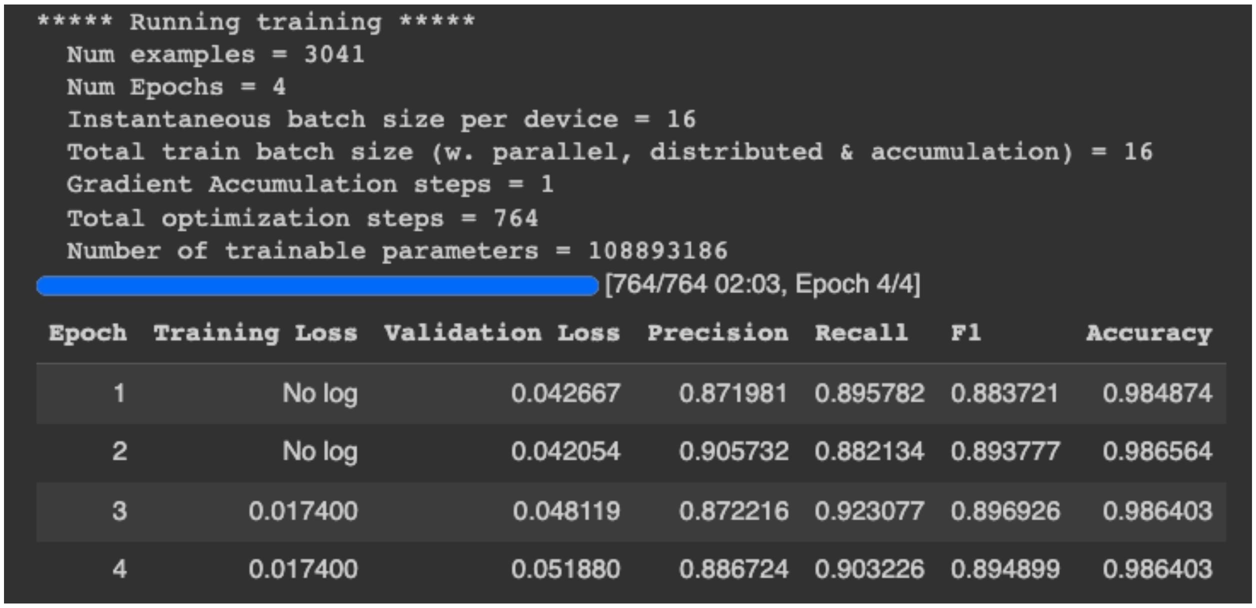
The high-level pipeline involves two major components which are the ABSA task and Recommender systems. The workflow is shown below:



Aspect Based Sentiment Analysis
Recommender Systems
Reviews
Aspect extraction
Aspect oriented Opinion word mining
Sentiment Analysis
Query Products based on required aspects
Reweighting candidate items
Top N Recommendations

6.1. ABSA – ATE Task

Aspect term extraction (ATE) is the first sub-problem of the ABSA task. The training notebook for the ATE task can be found here. We begin by using the *bert-base-uncased* model and training the ATE model as a sequence tagging task. Once the model is trained we evaluate on the test data set. The output of the ATE model is to predict the aspect terms given a text. The evaluation metrics of the trained ATE model are shown below.

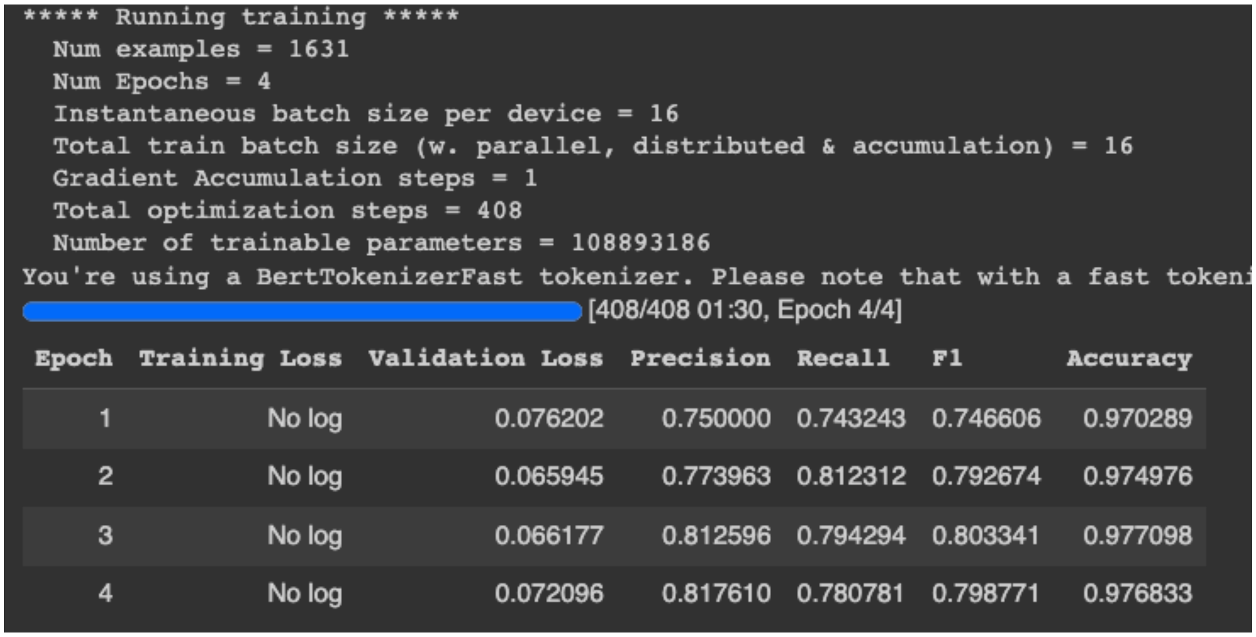


```
***** Running training *****
  Num examples = 3041
  Num Epochs = 4
  Instantaneous batch size per device = 16
  Total train batch size (w. parallel, distributed & accumulation) = 16
  Gradient Accumulation steps = 1
  Total optimization steps = 764
  Number of trainable parameters = 108893186
[764/764 02:03, Epoch 4/4]
```

Epoch	Training Loss	Validation Loss	Precision	Recall	F1	Accuracy
1	No log	0.042667	0.871981	0.895782	0.883721	0.984874
2	No log	0.042054	0.905732	0.882134	0.893777	0.986564
3	0.017400	0.048119	0.872216	0.923077	0.896926	0.986403
4	0.017400	0.051880	0.886724	0.903226	0.894899	0.986403

6.2. ABSA – TOWE Task

Target-oriented opinion word extraction (TOWE) is the next subtask of the ABSA task. This model aims to extract the opinion word associated with each aspect term. We use the *'bert-base-uncased'* model and train the model as a sequence tagging task. To infuse the target aspect term to the input review text, we add an additional prompt as discussed in section 5.2. The output of the TOWE model is given a <review, aspect term> tuple, which predicts the corresponding opinion word of the aspect term from the review text. The training notebook can be found here and the evaluation output of the TOWE mode is shown below.



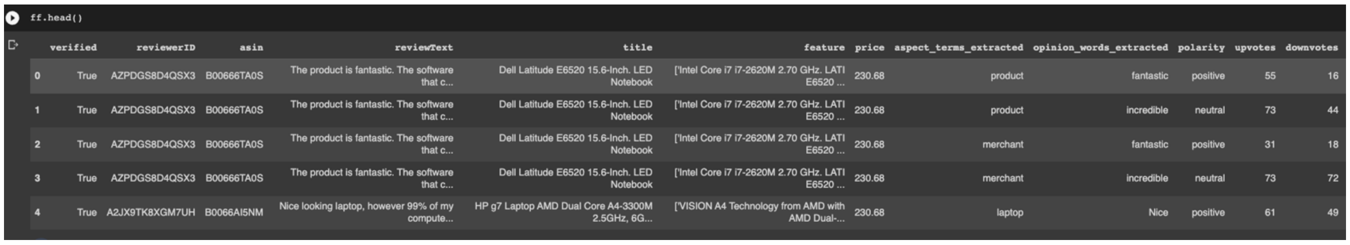
```
***** Running training *****
  Num examples = 1631
  Num Epochs = 4
  Instantaneous batch size per device = 16
  Total train batch size (w. parallel, distributed & accumulation) = 16
  Gradient Accumulation steps = 1
  Total optimization steps = 408
  Number of trainable parameters = 108893186
You're using a BertTokenizerFast tokenizer. Please note that with a fast tokeni
                                                 [408/408 01:30, Epoch 4/4]
```

Epoch	Training Loss	Validation Loss	Precision	Recall	F1	Accuracy
1	No log	0.076202	0.750000	0.743243	0.746606	0.970289
2	No log	0.065945	0.773963	0.812312	0.792674	0.974976
3	No log	0.066177	0.812596	0.794294	0.803341	0.977098
4	No log	0.072096	0.817610	0.780781	0.798771	0.976833

6.3. Recommender System – Query Products

We utilize the models that were used to extract aspects (ATE model) and extract opinion words corresponding to aspects (TOWE model). The final data frame created as described in section 5.3 is used. Using the <review, aspect> pair first we remove those pairs where the aspect term is not a noun or noun phrase (NN, NNP) using Nltk parts of speech tags since aspects are nouns. After applying this filter, the corresponding opinion word is extracted for each pair. We use the <review, aspect, opinion word> tuple and remove those tuples where the opinion word is not an adjective (ADJ) using the Nltk parts of speech tag. The final filtered tuples are passed to the sentiment analysis model to extract the polarity of the opinion words using the VADER (Valence Aware Dictionary of Sentiment Reasoning).

This final dataset with tuples <review id, review, aspect, opinion word, polarity> is joined to the final data frame created in section 5.3. The merged data frame M has tuples of the form <review id, review, aspect, opinion word, polarity, product id, product id, price, upvotes, downvotes>. We use the additional product-based features to extract the candidate terms. The final data M is shown below.



```
ff.head()
```

	verified	reviewerID	asin	reviewText	title	feature	price	aspect_terms_extracted	opinion_words_extracted	polarity	upvotes	downvotes
0	True	AZPDGS8D4QSX3	B00666TA0S	The product is fantastic. The software that c...	Dell Latitude E6520 15.6-Inch. LED Notebook	['Intel Core i7 i7-2620M 2.70 GHz. LATI E6520 ...	230.68	product	fantastic	positive	55	16
1	True	AZPDGS8D4QSX3	B00666TA0S	The product is fantastic. The software that c...	Dell Latitude E6520 15.6-Inch. LED Notebook	['Intel Core i7 i7-2620M 2.70 GHz. LATI E6520 ...	230.68	product	incredible	neutral	73	44
2	True	AZPDGS8D4QSX3	B00666TA0S	The product is fantastic. The software that c...	Dell Latitude E6520 15.6-Inch. LED Notebook	['Intel Core i7 i7-2620M 2.70 GHz. LATI E6520 ...	230.68	merchant	fantastic	positive	31	18
3	True	AZPDGS8D4QSX3	B00666TA0S	The product is fantastic. The software that c...	Dell Latitude E6520 15.6-Inch. LED Notebook	['Intel Core i7 i7-2620M 2.70 GHz. LATI E6520 ...	230.68	merchant	incredible	neutral	73	72
4	True	A2JX9TK8XGM7UH	B0066AI5NM	Nice looking laptop, however 99% of my compute...	HP g7 Laptop AMD Dual Core A4-3300M 2.5GHz, 6G...	['VISION A4 Technology from AMD with AMD Dual-...	230.68	laptop	Nice	positive	61	49

When the user enters a query to get the top N laptops with a certain feature *f* such as display, the query function filters all products with aspects similar to feature *f* in the aspects column. This first filter represents the set of candidate items C.

6.4 Recommender System – Candidate Reweighting

Now the problem with the dataset C is to recommend the top N items from a set C where N ⊆ C. Thus, we have to assign weights to the candidate items C. This is done using simple mathematical heuristics which we call as the *helpfulness score (h)*. The metric h is defined as follows:

Where,

(Review Helpfulness)

This function considers the effect of the positive and negative reviews and the helpfulness of each review which is the proportion of upvotes each review received. The helpfulness scores can be intuitively thought of as subtracting the influence of negative reviews (# of negative reviews) weighted by the average review helpfulness of all negative reviews for a product from the influence of positive reviews (# of positive reviews) weighted by the helpfulness of all positive reviews for the product in consideration. This difference is normalized by the count of all reviews. The value of h ∈ [−1, 1], where −1 indicates the product has more negative experience than positive. A value of 0 indicates that the experience was neutral and a value greater than 0 would indicate that the experience was positive for other users. The code for the following is shown below:

```
def calculate_weighted_confidence(dataframe):
    conf_list = []
    for gdf in dataframe.groupby('asin'):
        total_reviews = gdf[1].shape[0]
        num_pos_reviews = gdf[1][gdf[1]['polarity'] ==
'positive'].shape[0]
        num_neg_reviews = gdf[1][gdf[1]['polarity'] ==
'negative'].shape[0]
        num_neut_reviews = gdf[1][gdf[1]['polarity'] ==
'neutral'].shape[0]
        mean_conf_pos = gdf[1][gdf[1]['polarity'] ==
'positive']['confidence'].mean()
        mean_conf_neg = gdf[1][gdf[1]['polarity'] ==
'negative']['confidence'].mean()
        mean_conf_neut = gdf[1][gdf[1]['polarity'] ==
'neutral']['confidence'].mean()
        weighted_confidence = (num_pos_reviews*mean_conf_pos)-
(num_neg_reviews*mean_conf_neg)
        if str(weighted_confidence) == 'nan':
            if str((num_pos_reviews*mean_conf_pos)) != 'nan' and
num_pos_reviews>=2:
                weighted_confidence = num_pos_reviews*mean_conf_pos
            else:
                weighted_confidence = 0
        conf_list.append(weighted_confidence/total_reviews)
    return conf_list
```

Once the candidate items C, are weighted, the query function can choose the items $c_i \in C$, which are positive viz. greater than 0. These filtered candidates are sorted, and the top N candidates are recommended. The code for the query function is shown below.

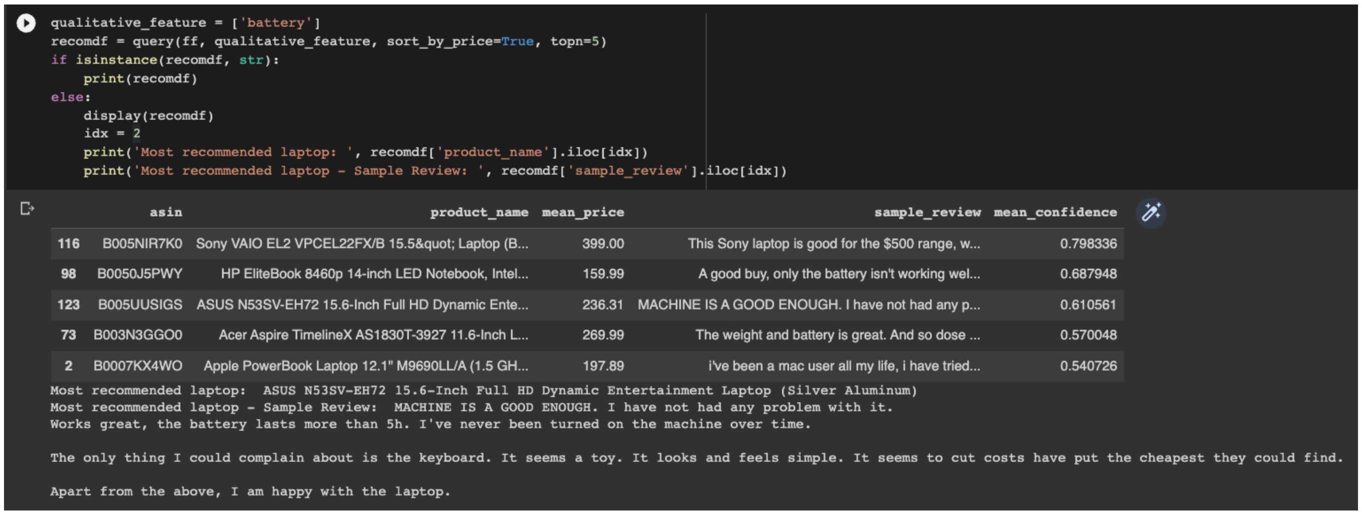
```
def query(dataframe, feature, sort_by_price = True, topn = 5, thresh =
0.4):
    qdf = dataframe[(dataframe['aspect_terms_extracted'].isin(feature))]
    weighted_confidence = calculate_weighted_confidence(qdf)
    qdf = qdf.groupby('asin').agg({'title':'first', 'price': 'mean',
'reviewText':'first'})
    qdf['weighted_conf'] = weighted_confidence
    qdf.columns = ['product_name', 'mean_price', 'sample_review',
'mean_confidence']
    qdf = qdf.reset_index()
    if sort_by_price:
        qdf = qdf.sort_values(by = ['mean_confidence', 'mean_price'],
ascending=[False, False])
    else:
        qdf = qdf.sort_values(by = ['mean_confidence', 'mean_price'],
ascending=[False, True])
    qdf = qdf[qdf['mean_confidence']>thresh]
    if qdf.shape[0] == 0:
        return 'No good recommendation found'
    return qdf.nlargest(topn, columns = 'mean_confidence')
```

The code for the final pipeline can be found here at this URL.

7. Results

The outputs of few queries are shown in this section. From the experimental evaluation it is seen that the results are convincing. Since the recommender system tasks do not have a labelled ground truth, the only way to evaluate its true measure is manual testing and evaluate feedback from real users after deploying it in production. Since that is out of scope for this project, we present some of the manual experimental outputs.

A) The initial query was for the implicit term battery. This would signify the user is interested in laptops that have along battery life. The query function returns the top 5 laptops where users have had a good experience with battery life. A sample review can also be seen to evaluate the performance. The user explicitly mentions *“The battery lasts more than 5h”*.

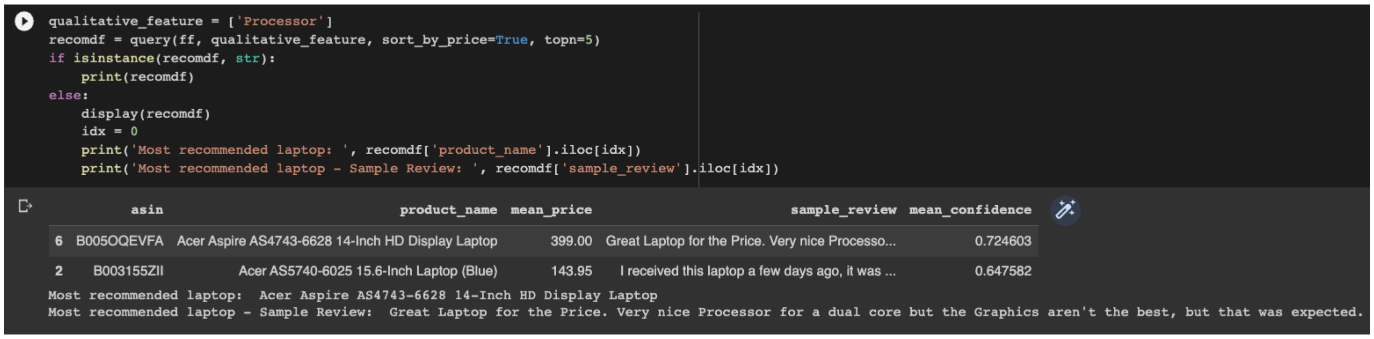


```python
qualitative_feature = ['battery']
recomdf = query(ff, qualitative_feature, sort_by_price=True, topn=5)
if isinstance(recomdf, str):
    print(recomdf)
else:
    display(recomdf)
    idx = 2
    print('Most recommended laptop: ', recomdf['product_name'].iloc[idx])
    print('Most recommended laptop - Sample Review: ', recomdf['sample_review'].iloc[idx])
```

	asin	product_name	mean_price	sample_review	mean_confidence
116	B005NIR7K0	Sony VAIO EL2 VPCEL22FX/B 15.5&quot; Laptop (B...	399.00	This Sony laptop is good for the $500 range, w...	0.798336
98	B0050J5PWY	HP EliteBook 8460p 14-inch LED Notebook, Intel...	159.99	A good buy, only the battery isn't working wel...	0.687948
123	B005UUSIGS	ASUS N53SV-EH72 15.6-Inch Full HD Dynamic Ente...	236.31	MACHINE IS A GOOD ENOUGH. I have not had any p...	0.610561
73	B003N3GGO0	Acer Aspire TimelineX AS1830T-3927 11.6-Inch L...	269.99	The weight and battery is great. And so dose ...	0.570048
2	B0007KX4WO	Apple PowerBook Laptop 12.1" M9690LL/A (1.5 GH...	197.89	i've been a mac user all my life, i have tried...	0.540726

```
Most recommended laptop:  ASUS N53SV-EH72 15.6-Inch Full HD Dynamic Entertainment Laptop (Silver Aluminum)
Most recommended laptop - Sample Review:  MACHINE IS A GOOD ENOUGH. I have not had any problem with it.
Works great, the battery lasts more than 5h. I've never been turned on the machine over time.

The only thing I could complain about is the keyboard. It seems a toy. It looks and feels simple. It seems to cut costs have put the cheapest they could find.

Apart from the above, I am happy with the laptop.
```

B) The second query was for the feature *“Processor”*. The output returned by the system is shown below. The user explicitly mentions *“ Very nice processor for the dual core”*.



```python
qualitative_feature = ['Processor']
recomdf = query(ff, qualitative_feature, sort_by_price=True, topn=5)
if isinstance(recomdf, str):
    print(recomdf)
else:
    display(recomdf)
    idx = 0
    print('Most recommended laptop: ', recomdf['product_name'].iloc[idx])
    print('Most recommended laptop - Sample Review: ', recomdf['sample_review'].iloc[idx])
```

	asin	product_name	mean_price	sample_review	mean_confidence
6	B005OQEVFA	Acer Aspire AS4743-6628 14-Inch HD Display Laptop	399.00	Great Laptop for the Price. Very nice Processo...	0.724603
2	B003155ZII	Acer AS5740-6025 15.6-Inch Laptop (Blue)	143.95	I received this laptop a few days ago, it was ...	0.647582

```
Most recommended laptop:  Acer Aspire AS4743-6628 14-Inch HD Display Laptop
Most recommended laptop - Sample Review:  Great Laptop for the Price. Very nice Processor for a dual core but the Graphics aren't the best, but that was expected.
```

8. Conclusion:

It can be seen that the recommender system works adequately for most queries. The true test of the system would be direct user feedback once it is deployed. We believe this project answers the question and is solving a real world problem of recommending products based on implicit and qualitative features. Further scope would be extrapolating the system scope to other domains which will have widespread applications

References

Chen, L., Chen, G., & Wang, F. (2015). Recommender systems based on user reviews: The state of the art. *User Modeling and User-Adapted Interaction*, *25*(2), 99–154. https://doi.org/10.1007/s11257-015-9155-5

Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up? *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - EMNLP '02*. https://doi.org/10.3115/1118693.1118704

Hu, M., & Liu, B. (2004). Mining and summarizing customer reviews. *Proceedings of the 2004 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '04*. https://doi.org/10.1145/1014052.1014073

Pontiki, M., Galanis, D., Pavlopoulos, J., Papageorgiou, H., Androutsopoulos, I., & Manandhar, S. (2014). Semeval-2014 task 4: Aspect based sentiment analysis. *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. https://doi.org/10.3115/v1/s14-2004

Fan, Z., Wu, Z., Dai, X.-Y., Huang, S., & Chen, J. (2019). Target-oriented opinion words extraction with Target-fused neural sequence labeling. *Proceedings of the 2019 Conference of the North*. https://doi.org/10.18653/v1/n19-1259

Ni, J., Li, J., & McAuley, J. (2019). Justifying recommendations using distantly-labeled reviews and fine-grained aspects. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. https://doi.org/10.18653/v1/d19-1018

Che, W., Zhao, Y., Guo, H., Su, Z., & Liu, T. (2015). Sentence compression for aspect-based sentiment analysis. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *23*(12), 2111–2124. https://doi.org/10.1109/taslp.2015.2443982

Wang, W., Pan, S. J., Dahlmeier, D., & Xiao, X. (2016). Recursive neural conditional random fields for aspect-based sentiment analysis. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. https://doi.org/10.18653/v1/d16-1059

Akhtar, M. S., Gupta, D., Ekbal, A., & Bhattacharyya, P. (2017). Feature selection and Ensemble Construction: A two-step method for aspect-based sentiment analysis. *Knowledge-Based Systems*, *125*, 116–135. https://doi.org/10.1016/j.knosys.2017.03.020

Yang, C., Zhang, H., Jiang, B., & Li, K. (2019). Aspect-based sentiment analysis with alternating coattention networks. *Information Processing & Management*, *56*(3), 463–478. https://doi.org/10.1016/j.ipm.2018.12.004

Wang, K., Shen, W., Yang, Y., Quan, X., & Wang, R. (2020). Relational graph attention network for aspect-based sentiment analysis. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. https://doi.org/10.18653/v1/2020.acl-main.295

Liu, M. Z., Zhou, F. Y., Chen, K., & Zhao, Y. (2021). Co-attention networks based on aspect and context for aspect-level sentiment analysis. *Knowledge-Based Systems*, *217*, 106810. https://doi.org/10.1016/j.knosys.2021.106810