Curience Forecasting Intern Report

Github Repo Link:
https://github.com/pallavpp/Curience-Work

Spreadsheet Link:
https://docs.google.com/spreadsheets/d/18Q0NVDeW2YqVaxJFyY8bSKJCIWaMHAuO5oYcWqDYeQg/edit?usp=sharing

To view this document as a google doc:
https://docs.google.com/document/d/1RS7Ipab4UIDa-klV3QauxywEPghJm06_RBgKH31ZHZQ/edit?usp=sharing

Task 1: Scraping and Data Collection

The aim is to create a high recall set.
Websites Scraped:

- Glamour Magazine UK : https://www.glamourmagazine.co.uk/topic/fashion
- Instyle fashion : https://www.instyle.com/fashion/clothing
- Purewow fashion : https://www.purewow.com/fashion
- TheVou fashion : https://thevou.com/fashion/
- Vogue India : https://www.vogue.in/fashion

Websites like Harper's bazaar, whowhatwear, etc., should be included in the future.

Following fields were obtained through various blogs -
["Blog Title", "Blog Date", "Blog Catchphrase", "Blog Category", "Blog Link", "Author Name", "Author Profile Link", "Thumbnail Link", "Thumbnail Credit"]

Static / Dynamic scraping:

- Static scraping is done using beautiful soup.
- Dynamic scraping is done using selenium and xpath.

All site-specific scraping scripts can be found in the 'Scripts' folder:
https://github.com/pallavpp/Curience-Work/tree/main/Scripts

All site-specific scraped data can be found in the 'Data' folder or in the Spreadsheet Link:
https://github.com/pallavpp/Curience-Work/tree/main/Data

Task 2: Accumulating the Data

Currently aiming for - fashion/trend forecast/prediction spring/summer 2022

The following keywords are used for naive shortlisting -
["fashion", "trend", "forecast", "prediction", "spring", "summer", "2022"]
Each blog gets a keyword score (number of keywords occurring in the title). All scraped data is then combined and sorted in decreasing order of keyword count.
This data can be viewed in the 'Filtered_Data' folder or in the Spreadsheet Link with the name 'cumulative_data_with_keyword_count.csv':
https://github.com/pallavpp/Curience-Work/blob/main/Filtered_Data/cumulative_data_with_keyword_count.csv

Ngrams were looked for in blog titles of blogs with a non-zero score to check if we have missed any terms related to our keywords. No additional useful terms were found.
Outputs for the same can be viewed in the 'Filtered_Data/Ngram_Histogram_Plots' folder:
https://github.com/pallavpp/Curience-Work/tree/main/Filtered_Data/Ngram_Histogram_Plots

Final shortlisting is done using a scoring criteria. The following set of rules is used to score the documents and sort them in the decreasing order of score:

- Start with an initial score of 0
- If some year is present in the Blog Title - remove if >= 2010 and <= 2021. If 2022 is present, the blog can't be removed in future steps.
- If Blog Date or Blog/Thumbnail Link contains year < 2021 - remove if permitted.
- +1 for each unique keyword present in the title
- +1 additionally for ("summer" or "spring" or "2022") presence in title
- +1 additionally for "2022" presence in the blog link

The top 100 blogs obtained via this scoring will be our final dataset to be worked upon.
This data can be viewed in the 'Filtered_Data' folder or in the Spreadsheet Link with the name 'final_data.csv':
https://github.com/pallavpp/Curience-Work/blob/main/Filtered_Data/final_data.csv

Source diversity does not seem to be a problem.
Some false data such as "Summer 21" is present among the top results. These specific scenarios can be dealt with by hard coding them but are not done yet as these are few in number.
Website penetration can be used as a factor in scoring in the future. It can be estimated by using google trends. We can use google to cross-validate our top data.

The golden set considered is the list of blogs provided by the fashion intern. It can be viewed in the 'Read_Files' folder and has the name 'fashion_intern_forecasting_website_list.csv'. It can also be viewed in the Spreadsheet Link with the name 'List of websites from fashion intern': https://github.com/pallavpp/Curience-Work/blob/main/Read_Files/fashion_intern_forecasting_website_list.csv

Recall analysis -
A total of 3 blogs were common among the cumulative data and the fashion intern data. A total of 5 websites have been scraped, and among those, only 3 links were provided by the fashion intern. We're getting all of them from the cumulative data.
This scenario tells us that we should scrap some other websites as well, especially the ones from which the fashion intern has listed a larger number of blogs, such as vogue.

Precision analysis -
In our final rankings, out of these 3 links, two links are at positions 3 and 6, and one is at position 81. The reason for the 81st position is that there were no keywords in the blog title.
Assuming we include the 20 websites provided by fashion intern:
    Almost 50% of those links would have been in the top 30 of our final list.
Almost 60% of those links would have been in the top 50 of our final list.
Scoring seems to be reasonable for now.

Task 3: Document analytics

We have 100 shortlisted blogs. Pallav and Alokeveer tested their methods on a set of 50 blogs. For individual testing, Pallav used the set of odd rows from the final data, Alokeveer used the set of even rows from the final data.

We have to extract those features based on which we can extract info. Following techniques were tested:

Pallav:

- Detecting nouns from ngram frequency
- Tf-idf scoring
- RAKE algorithm

Complete precision analysis can be viewed in the Spreadsheet Link with the name 'precision_analysis_bigrams'. Precisions for all techniques are calculated manually on the obtained outputs for bigrams only.

Alokeveer:

- Finding prefix and suffix for fashion-related nouns
- Important sentence detection using vocabulary matching

Textrank algorithm was tested on a blog. While the results obtained from running only textrank were subpar, many research papers on 'Information Retrieval' claimed that results are better when techniques are run on abstracts rather than full text.
Textrank can be used to create a summary for each blog, and all techniques can then be tested on these summaries.

Pallav's Observations

Tfidf is a corpus-dependent method and captures the essence of the entire corpus. This is better for our task in comparison to rake, as rake is an individual document-oriented information retrieval method. But, tfidf is very costly, whereas rake is one of the most efficient algorithms out there.

For fashion terms, 'cumulative tfidf' gave the best results. 'Cumulative tfidf' also gave the best overall precision.
For detection of nouns from ngarm frequency, blogwise_rake was able to provide equal results.
'Cumulative tfidf' is good for detecting fashion-related terms, as almost all of its score is from fashion terms. It was able to detect some fashion terms not detected in any other method.
'Cumulative rake' gave decent results, but this seems to be a fluke, as the output was not good when viewed manually.

All these methods are explained below.

1. <u>Detecting nouns from ngram frequency</u>:
An ngram is considered to be a noun if it contains a capital letter. The top 20 nouns are found in each blog. A noun occurring in the top 20 of more than one blog is considered for summary. The number of blogs it occurs in is the term's occurrence count.
The output can be viewed in the 'Analytics/Pallav/Outputs' folder with name 'nouns_from_ngrams.txt':
https://github.com/pallavpp/Curience-Work/blob/main/Analytics/Pallav/Outputs/nouns_from_ngrams.txt
NER models could be used in future to improve detection.

Precision Analysis:
The summary obtained was used for precision analysis. Precision was calculated manually.
Fashion terms - 8%
People/Names - 13.33%
Brands/Other Nouns - 8%
Cumulative - 29.33%

2. <u>Tf-idf scoring</u>:
Two methods are used. Firstly, the top 20 scoring ngrams are taken from each blog. Ngrams occurring in the top 20 of more than one blog are chosen for the summary. The number of blogs it occurs in is the terms occurrence count.

Outputs for the same can be viewed in the 'Analytics/Pallav/Outputs' folder with the name 'blogwise_tfidf.txt':
https://github.com/pallavpp/Curience-Work/blob/main/Analytics/Pallav/Outputs/blogwise_tfidf.txt

Precision Analysis (blogwise tfidf):
The summary obtained was used for precision analysis. Precision was calculated manually.
Fashion terms - 15.90%
People/Names - 9.09%
Brands/Other Nouns - 3.40%
Cumulative - 28.41%

In the next method, cumulative scoring is done. Each occurrence of a word will have various scores (each score corresponding to a blog it is occurring in). The maximum score found for a word is considered as its score. All words are then merged and sorted based on this score. The top 50 words are considered.
Outputs for the same can be viewed in the 'Analytics/Pallav/Outputs' folder with the name 'cumulative_tfidf.txt:
https://github.com/pallavpp/Curience-Work/blob/main/Analytics/Pallav/Outputs/cumulative_tfidf.txt

Precision Analysis (cumulative tfidf):
The summary obtained was used for precision analysis. Precision was calculated manually.
Fashion terms - 36%
People/Names - 2%
Brands/Other Nouns - 8%
Cumulative - 46%

3. RAKE algorithm:
Two methods are used. Firstly, the top 20 scoring ngrams are taken from each blog. Ngrams occurring in the top 20 of more than one blog are chosen for the summary. The number of blogs it occurs in is the terms occurrence count.
Outputs for the same can be viewed in the 'Analytics/Pallav/Outputs' folder with the name 'blogwise_rake.txt':
https://github.com/pallavpp/Curience-Work/blob/main/Analytics/Pallav/Outputs/blogwise_rake.txt

Precision Analysis (blogwise rake):
The summary obtained was used for precision analysis. Precision was calculated manually.
Fashion terms - 19.67%
People/Names - 13.11%
Brands/Other Nouns - 1.64%
Cumulative - 34.42%

In the next method, all blogwise text is merged. Rake is then applied on this complete corpus at once. Top 100 words are considered.
Outputs for the same can be viewed in the 'Analytics/Pallav/Outputs' folder with the name 'cumulative_rake.txt:
https://github.com/pallavpp/Curience-Work/blob/main/Analytics/Pallav/Outputs/cumulative_rake.txt

Precision Analysis (cumulative rake):
The summary obtained was used for precision analysis. Precision was calculated manually.
Fashion terms - 25%
People/Names - 1%
Brands/Other Nouns - 3%
Cumulative - 29%

Alokeveer’s Observations

The fashion intern provided us with a fashion vocabulary.
This data can be viewed in the 'Read_Files' folder with the name 'fashion_vocabulary.csv' or in the Spreadsheet Link with the name 'Fashion Vocabulary': https://github.com/pallavpp/Curience-Work/blob/main/Read_Files/fashion_vocabulary.csv

1. Finding prefix and suffix for fashion-related nouns:
The result obtained from this method can be used for vocabulary building.
The specifications column of the table fashion_vocabulary.csv is read, and the last word from each row is treated as a noun. Two approaches are taken from here:
   (a) For individual blogs
   (b) For all the blogs combined

In both cases, the whole set of words is traversed. For each noun, their prefixes and suffixes with the frequencies are stored and sorted in descending order. The top 3 prefixes and suffixes for each noun are finally printed. The outputs can be viewed here:
   (a) For individual blogs:
      https://github.com/pallavpp/Curience-Work/blob/main/Analytics/Alokeveer/Outputs/blogwise_vocab_match_prefix_suffix.txt
   (b) For all the blogs combined:
      https://github.com/pallavpp/Curience-Work/blob/main/Analytics/Alokeveer/Outputs/cumulative_vocab_match_prefix_suffix.txt

Some interesting observations are visible like :
Prefix - "pleated", noun - "skirt"
Noun - "biker", suffix - "jacket"
Noun- "hemilene", suffix - "raised"
Prefix - “popcorn”, noun - “tops” etc.

2. Important sentence detection using vocabulary matching:
The nouns for this task are the same as the previous one. An adjective is a word preceding the noun in the specification column of fashion vocabulary. All sentences are extracted and stored in a set. Now the following algorithm is used:
For each noun, we traverse through each sentence in the set of sentences. If the noun is not present in the sentence, we neglect the sentence. Otherwise, we do the following:

(a) Consider the adjective of the noun from the fashion vocabulary table.
(b) Search the adjective near the position of the noun in the current sentence (3 words before the position of the noun and 3 words after the position of the noun).
(c) If the adjective is found, the sentence is useful for us. Otherwise, we neglect the sentence.

In total, when all the blogs were combined, and this algorithm was run over all the sentences, 102 sentences were obtained, and by manual reading, all of them seemed to be useful for forecasting.

The output can be viewed here:
https://github.com/pallavpp/Curience-Work/blob/main/Analytics/Alokeveer/Outputs/cumulative_vocab_match_sentences.txt

Future Improvements

1. Corpus-dependent methods should be preferred. Other analytic techniques that can be tested:

- YAKE algorithm
- KeyBERT (Very promising)
- Semantic methods for keywords (Eg. GenEx)
- GPT-3

2. Finding text corresponding to each img on the page (text can be used as caption/description for the image). The tree structure of the page can be used for locating text.
Some useful links to start with:

    Text nearest to img -
    https://stackoverflow.com/questions/34270214/find-the-text-nearest-to-an-element-in-html-content
    Common Ancestor -
    https://stackoverflow.com/questions/3960843/how-to-find-the-nearest-common-ancestors-of-two-or-more-nodes

3. Source-based filtering can be applied while accumulating output:

- No of sources in the output
- Particular source occurrence in top k results
- Setting a minimum score criteria

4. Add google search results to the blog list (risky)
5. Use a neural network for summarization

Note: Check ReadMe for better understanding of the code.