CSCE625: Artificial Intelligence

Communication Project

Final Report

Prof. Dylan Shell

Recommending Fragrance ingredients using Search Algorithms

Abdullatif AlNuaimi

73100942

Introduction:

Nowadays, fragrances have improved the grooming habits of individuals and have become essential products of day-to-day life. According to a study from Mordor Intelligence (https://www.mordorintelligence.com/industry-reports/fragrance-and-perfume-market) that the market is growing rapidly from 2022 to 2027 with a yearly grow ratio of 5.52 % and a market capitalization of 52.4 billion U.S. dollars as shows in the below figure. This is because of the continuous effort on research and development among companies worldwide especially in fragrance creation since it is the starting point of any successful perfume. The people who are responsible for fragrance creation known as perfumers.

Who is a Perfumer?

Perfumers are trained professional artists who create fragrances for a multitude of products. Artistic as well as scientific, perfumers' study long hours and many years to achieve the sensitive balance required to create just the right scents. The training process to become a perfumer is mentally intense, rigorous, and lengthy. The first task for a trainee perfumer is to study hundreds of natural and synthetic materials, memorizing them according to their specific odor and category. Once a mastery of this olfactory memory is attained, one must be able to create both simple and complex fragrance accords. By combining, balancing, and blending these fragrance ingredients in memorable and unique ways, new fragrances are developed. Much like a musician working with notes and chords, infinite possibilities can be expressed.

What are fragrance ingredients?

Fragrance is defined by the Food and Drug Administration (FDA) as a combination of chemicals that gives each perfume or cologne (including those used in other products) its distinct scent. Fragrance ingredients may be derived from petroleum or natural raw materials including Essential oils and Aroma Chemicals.

What are essential oils?

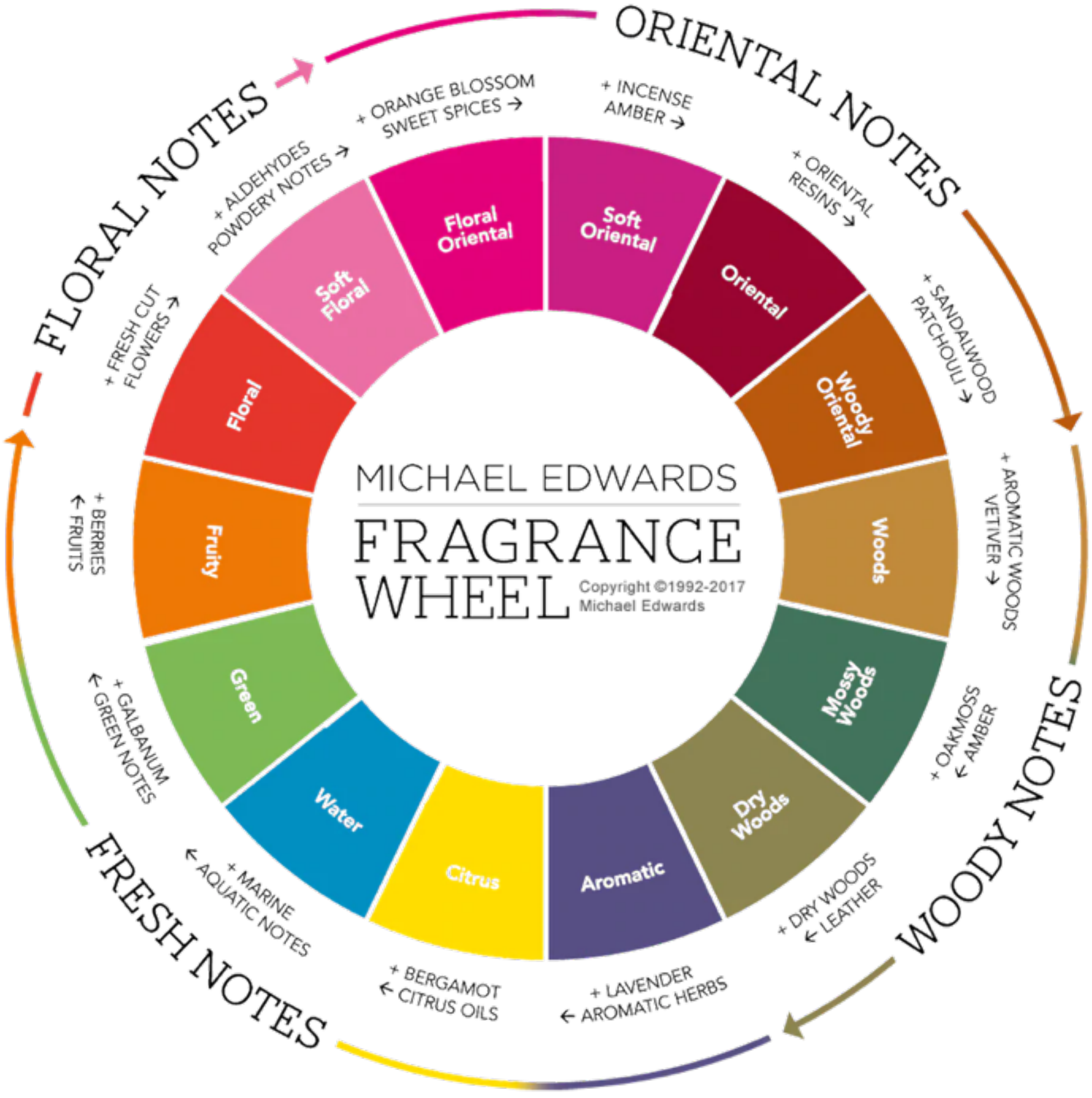
Essential oils are compounds extracted from plants. The oils capture the plant's scent and flavor, or "essence." Unique aromatic compounds give each essential oil its characteristic essence. Essential oils are obtained through distillation (via steam and/or water) or mechanical methods, such as cold pressing. Once the aromatic chemicals have been extracted, they are combined with a carrier oil to create a product that's ready for use.

What are aroma chemicals?

Aroma chemicals are compounds possessing a complex and distinct scent that enhances the aroma or fragrance of formulations they are infused in. These aroma chemicals are highly volatile with the ability to easily disperse any scent, which enhances the diffusive properties required to create long-lasting fragrances. Aroma chemicals can be used to infuse fragrance into various products with their simple, pure, and manageable aroma. Along with Essential Oils, aroma chemicals form the backbone of ingredients used when formulating flavors and fragrances.

How to describe fragrance scent?

A common way to describe a scent is through fragrance wheel. It is a round diagram that displays the different scent families and subfamilies. The scents are grouped based on their similarities and differences to show their relationship to one another. The scent groups that border each other share common olfactory characteristics, while those that are further away from one another are less related. This fragrance classification system was developed by fragrance expert Michael Edwards in order to help retailers suggest perfumes to consumers more efficiently. Each family consists of a prominent scent, while the subfamilies are blended versions of these fragrances.



Problem Statement & Motivation:

Since I'm a certified perfumer, I want to address two real-life problems that perfumers face constantly which are:

1. No accessible database of fragrance ingredients
2. Replace the traditional way, trial & error, of choosing suitable fragrance ingredients

My approach:

The idea is to build a trusted database and that will feed into an advisory system to recommend which fragrance ingredients to use based on perfumer's input using searching algorithm. My approach is divided in two parts:

The first part: Building a database

Having a trustworthy website that provide detailed information about fragrance ingredients is really challenging. My approach is to write a code that scrap the data and search for specific information from multiples well-known websites and store them in an excel file. To do this, I will use Scrapy framework in python which is large scale web scraping tool to efficiently extract data from websites. Then, I will do the data cleaning process and preparation for the next part.

The second part: Design fragrance ingredients' recommender system

The idea is to design a recommender system based on searching algorithms that helps perfumers to decide which ingredients to use in order to have creative blends. The input of the system will be descriptive words (which are adjective words appear in the fragrance wheel mentioned earlier) of what the perfumer wants to achieve as a final scent. For example, flowery, woody, and musky. Then the system uses searching algorithms to scan the database looking for best ingredients that match what the perfumer is asking for. The output will be some recommended fragrance ingredients including essential oils and aroma chemicals.

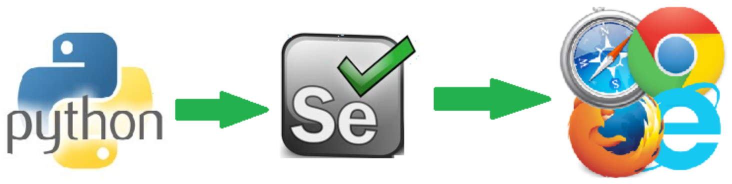
Data Collection Using Web Scrapping:

What is web scrapping?

Web scraping, harvesting, or data extraction are data scraping terminologies used for extracting data from websites. Web scraping a web page involves fetching it and extracting from it. Fetching is the downloading of a page (which a browser does when a user views a page). Therefore, web crawling is a main component of web scraping, to fetch pages for later processing. Once fetched, then extraction can take place. The content of a page may be parsed, searched, reformatted, its data copied into a spreadsheet or loaded into a database. Web scrapers typically take something out of a page, to make use of it for another purpose somewhere else. An example would be to find and copy names and telephone numbers, or companies and their URLs, or e-mail addresses to a list (contact scraping).

What tool I used?

I used Selenium to do web scraping. It is a Python library and tool used for automating web browsers to do a number of tasks. One of such is web-scraping to extract useful data and information that may be otherwise unavailable. It is a powerful web browser automation tool that can simulate operations that we humans like to do over the web. It extends its support to various browsers like Chrome, Internet Explorer, Safari, Edge, Firefox.



How I implement it?

I wrote my code on python to do web scraping on two trusted websites to collect data about essential oils (https://www.edensgarden.com/) and aroma chemicals (https://shop.perfumersapprentice.com/). The following columns of information gathered and saved in a separate excel sheets:

- **Essential oils (117 observations):** Ess_Oil_Name , Odor_Description, Perfumary_Note , Family, Ext_Method , and Blends_with
- **Aroma Chemicals (131 observations):** Aroma_Chemical, Odor_Description, Family, and Perfumary_Note

Essential oils web scrapping code:
https://colab.research.google.com/drive/1goTpV5mCJBWkFUvDNwmMhL8O7gnwQJmY?authuser=1#scrollTo=GlVDI4cEuG3t&line=59&uniqifier=1

Aroma Chemicals web scrapping code:
https://colab.research.google.com/drive/1goTpV5mCJBWkFUvDNwmMhL8O7gnwQJmY?authuser=1#scrollTo=SItQhXrnYiHU&line=5&uniqifier=1

Data Cleaning and Building Database:

Data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset. When combining multiple data sources, there are many opportunities for data to be duplicated or mislabeled. If data is incorrect, outcomes and algorithms are unreliable, even though they may look correct. I applied data cleaning techniques and then I saved the updated information in a new spread sheets to act as the database of my project.

Implementation of Search Algorithms:

Searching Algorithms are designed to check for an element or retrieve an element from any data structure where it is stored. The following searching algorithms is used to scan the database built and act as the building blocks for my recommender system:

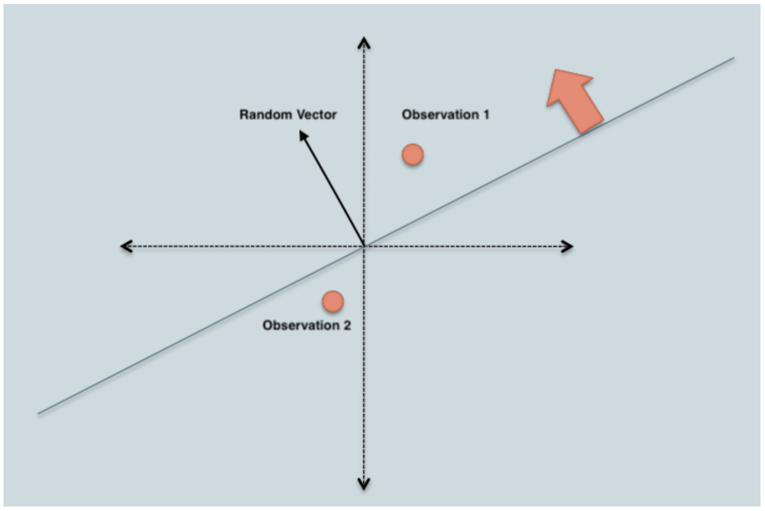
Approximate Nearest Neighbor (vector encoding using LSH)

Refers to a family of functions (known as LSH families) to hash data points into buckets so that data points near each other are in the same buckets with high probability, while data points far from each other are likely to be in different buckets. This makes it easier to identify observations with various degrees of similarity.

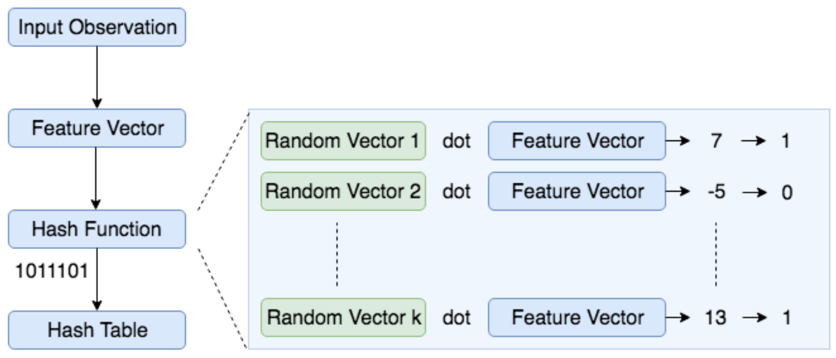
LSH primarily differs from conventional hashing (aka cryptographic) in the sense that cryptographic hashing tries to avoid collisions, but LSH aims to maximize collisions for similar points. I used here vector encoding using LSH.

How the model works?

In this LSH implementation, I construct a table of all possible bins where each bin is made up of similar items (descriptive words). Each bin will be represented by a bitwise hash value, which is a number made up of a sequence of 1's and 0's (Ex: 110110, 111001). In this representation, two observations with same bitwise hash values are more likely to be similar than those with different hashes. Basic algorithm to generate a bitwise hash table is



1. Create k random vectors of length d each, where k is the size of bitwise hash value and d is the dimension of the feature vector.
2. For each random vector, compute the dot product of the random vector and the observation. If the result of the dot product is positive, assign the bit value as 1 else 0
3. Concatenate all the bit values computed for k dot products
4. Repeat the above two steps for all observations to compute hash values for all observations
5. Group observations with same hash values together to create a LSH table



How I implement it?

I used Facebook Artificial Intelligence Similarity Search (FAISS) which is a Python library developed by Facebook Research that provides several built-in functions for implementing fast similarity search on CPU or GPU memory. Most of the similarity search methods from FAISS use a compressed representation of the data in order to speed up the search process. While this can lead to less precise results, the resulting payoff in memory storage and time saved can greatly out-weigh a marginal loss in search precision.

It assumes that the instances are represented as vectors and are identified by an integer, and that the vectors can be compared with L2 (Euclidean) distances or dot products. Vectors that are like a query vector are those that have the lowest L2 distance or the highest dot product with the query vector. It also supports cosine similarity since this is a dot product on normalized vectors.

Vector encoding using LSH code:
https://colab.research.google.com/drive/1G-fPoka3gr8OyvHv9J5QCsUTNf0-u_OH#scrollTo=HZN0Kdwa7Cmh&line=5&uniqifier=1

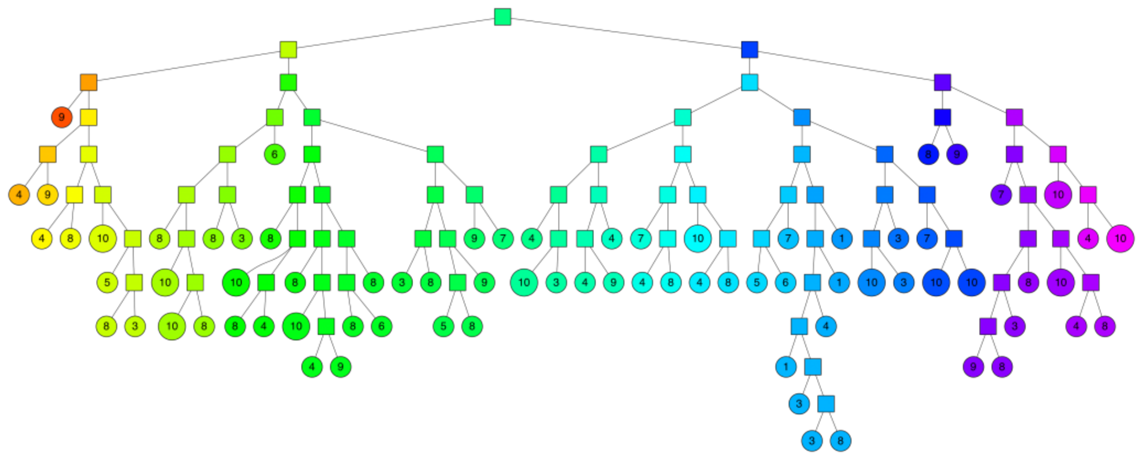
Approximate Nearest Neighbor (vector encoding using trees)

A form of proximity search is the optimization problem of finding the point in each set that is closest (or most similar) to a given point. Closeness is typically expressed in terms of a dissimilarity function: the less similar the objects, the larger the function values. Here I used vector encoding using trees.

Tree-based algorithms are one of the most common strategies when it comes to ANN. They construct forests (collection of trees) as their data structure by splitting the dataset into subsets.

One of the most prominent solutions is Annoy (Approximate Nearest Neighbors Oh Yeah), which uses trees (more accurately forests) to enable Spotify' music recommendations. In

Annoy, in order to construct the index we create a forest (aka many trees) Each tree is constructed in the following way, we pick two points at random and split the space into two by their hyperplane, we keep splitting into the subspaces recursively until the points associated with a node is small enough.



How I implement it?

I used Annoy which is a C++ library with Python bindings to search for points in space that are close to a given query point. It also creates large read-only file-based data structures that are MMAP (a POSIX-compliant Unix system call that maps files or device into memory) into memory so that many processes may share the same data. Annoy is almost as fast as the fastest libraries.

Another feature that really sets Annoy apart: it has the ability to use static files as indexes. This means you can share index across processes. Another nice thing of Annoy is that it tries to minimize memory footprint, so the indexes are quite small.

I'm going to create an index class. As you can imagine most of the logic is in the build method (index creation), where the accuracy-performance tradeoff is controlled by :

- **number_of_trees** — the number of binary trees we build, a larger value will give more accurate results, but larger indexes.
- **search_k** — the number of binary trees we search for each point, a larger value will give more accurate results, but will take a longer time to return.

Vector encoding using trees code:
https://colab.research.google.com/drive/1G-fPoka3gr8OyvHv9J5QCsUTNf0-u_OH#scrollTo=pEUDKS5f71Rd&line=5&uniqifier=1

Okapi Best Match 25 (BM25)

BM25 is a ranking function that ranks a set of documents based on the query terms appearing in each document, regardless of the inter-relationship between the query terms within a document (e.g., their relative proximity). It is not a single function, but actually a whole family of scoring functions, with slightly different components and parameters. It is used by search engines to rank matching documents according to their relevance to a given search query and is often referred to as "Okapi BM25," since the Okapi information retrieval system was the first system implementing this function.

The BM25 retrieval formula belongs to the BM family of retrieval models, that is the weight of a term t in a document d is:

$$\frac{\mathbf{tf}}{k+\mathbf{tf}} \ln \frac{(r_t+0.5)\cdot(\mathbf{N}-R-\mathbf{n}_t+r_t+0.5)}{(\mathbf{n}_t-r_t+0.5)\cdot(R-r_t+0.5)} [BM's\text{ family}]$$

Where:

- R is the number of documents known to be relevant to a specific topic,
- r t is the number of relevant documents containing the term,
- N is the number of documents of the collection,
- n t is the document frequency of the term,
- tf is the frequency of the term in the document,
- k is a parameter.

The BM25 document-query matching function is:

$$\sum_{t\in q} w_t \cdot \text{In} \frac{(r_t+0.5)\cdot(\mathbf{N}-R-\mathbf{n}_t+r_t+0.5)}{(\mathbf{n}_t-r_t+0.5)\cdot(R-r_t+0.5)} [BM25]$$

where

- $\mathbf{q}$ is the query,
- $\mathrm{w}_{\mathbf{t}} = (k_1 + 1)\frac{\mathbf{tf}}{k + \mathbf{tf}} \cdot (k_3 + 1)\frac{\mathbf{tf}_q}{(k_3 + \mathbf{tf}_\mathbf{q})}$
- $\mathbf{tf}_q$ is the frequency of the term within the topic from which $\mathbf{q}$ was derived
- l and $\bar{\mathrm{l}}$ are respectively the document length and average document length.
- k is $k_1 \left((1-b) + b(\frac{l}{\bar{\mathrm{l}}})\right)$.
- k_1, b and k_3 are parameters which depend on the nature of the queries and possibly on the database.
- k_1 and b are set by default to 1. 2 and 0. 75 respectively, k_3 to 1000.

By using these default parameters, the unexpanded BM25 ranking function, that is the BM25 applied in the absence of information about relevance (R = r = 0), is:

$$\sum_{\mathbf{t}\in\mathbf{q}} \frac{2.2 \cdot \mathbf{tf}}{0.3 + 0.9\frac{l}{\bar{\mathrm{l}}} + \mathbf{tf}} \cdot \frac{1001 \cdot \mathbf{tf}_q}{1000 + \mathbf{tf}_\mathbf{q}} \ln \frac{N - \mathbf{n}_\mathbf{t} + 0.5}{\mathbf{n}_\mathbf{t} + 0.5}$$

How I implement it?

Unlike other algorithms, I used BM25 like a bag-of-words retrieval function that ranks a set of documents based on the query terms appearing in each document, regardless of their proximity within the document. BM25 python package also can be used to index the data, descriptive words in our case, based on the search query. It works on the concept of TF/IDF i.e.

- TF or Term Frequency—Simply put, indicates the number of occurrences of the search term in our tweet
- IDF or Inverse Document Frequency—It measures how important your search term is. Since TF considers all terms equally important, thus, we can't only use term frequencies to calculate the weight of a term in your text. We would need to weigh down the frequent terms while scaling up the rare terms showing their relevancy to the descriptive word.

Okapi Best Match 25 code:
https://colab.research.google.com/drive/1G-fPoka3gr8OyvHv9J5QCsUTNf0-u_OH#scrollTo=8DOdqCqr8mxp&line=12&uniqifier=1

Results:

I have applied the previous algorithms on different scenarios:

First Scenario:

I have applied Approximate Nearest Neighbor (vector encoding using trees) on aroma chemical database to find similarities in existing observations i.e. between a specific description of one data point in the database with other data points. The input was index[3] with the following attribute " *fresh amber aldehydic moss citrus tuberose metallic waxy coumarinic* ". The output was the following:

```
Aroma_Chemical - Aldehyde C12 MNA
Perfume Family - Aldehydic
Perfume Odor_Strength - High
Perfumary_Note - Base
************
Aroma_Chemical - Aldehyde C11
Perfume Family - Aldehydic
Perfume Odor_Strength - High
Perfumary_Note - Base
************
Aroma_Chemical - Aldehyde C10 – Decanal
Perfume Family - Aldehydic
Perfume Odor_Strength - High
Perfumary_Note - Base
************
Aroma_Chemical - Aldehyde C12 Lauric
Perfume Family - Aldehydic
Perfume Odor_Strength - High
Perfumary_Note - Base
************
Aroma_Chemical - Ambrocenide Crystals (Symrise)
Perfume Family - Ambery
Perfume Odor_Strength - None
Perfumary_Note - Base
************
Aroma_Chemical - Fixateur 505
Perfume Family - Amber
Perfume Odor_Strength - Medium
Perfumary_Note - Base
************
Aroma_Chemical - Dihydromyrcenol
Perfume Family - Citrus
Perfume Odor_Strength - Medium
Perfumary_Note - Top
************
time taken 0.03936624526977539
```

Second Scenario:

I have applied Approximate Nearest Neighbor (vector encoding using LSH) on aroma chemical database to find similarities of an external fragrances description (attribute) with our observations in the database. The input was the following attribute “*mossy oakmoss woody phenolic earthy*”. The output was the following:

```
Aroma_Chemical - Farnesol
Perfume Family - Floral
Perfume Odor_Strength - Low
Perfumary_Note - Base
************
Aroma_Chemical - Melonal
Perfume Family - Melon
Perfume Odor_Strength - High
Perfumary_Note - Middle
************
Aroma_Chemical - Veramoss / Evernyl
Perfume Family - Mossy
Perfume Odor_Strength - High
Perfumary_Note - Base
************
Aroma_Chemical - Calone liquid
Perfume Family - Melon
Perfume Odor_Strength - High
Perfumary_Note - Base
************
Aroma_Chemical - Ethyl Maltol
Perfume Family - Caramellic
Perfume Odor_Strength - High
Perfumary_Note - Base
************
Aroma_Chemical - Aldehyde C12 MNA
Perfume Family - Aldehydic
Perfume Odor_Strength - High
Perfumary_Note - Base
************
Aroma_Chemical - Aldehyde C18 – gamma Nonalactone
Perfume Family - Coconut
Perfume Odor_Strength - Medium
Perfumary_Note - Base
************
time taken 0.03795218467712402
```

Third Scenario:

I have applied Okapi Best Match 25 on essential oils database to find similarities of a user input and do after that local search comparing the strings togethers and find at the end the best match. The user input was the following attribute "*fresh amber aldehydic moss citrus tuberose metallic waxy coumarinic*". The output was the following:

```
Oil name - Gingergrass Essential Oil
Perfumary_Note - Middle
Perfume family - Poaceae
Good for blend with - Citruses and florals such as Lavender
**********
Oil name - Neroli Essential Oil
Perfumary_Note - Middle
Perfume family - Rutaceae
Good for blend with - Frankincense, and other resins
**********
Oil name - Frankincense- Frereana Essential Oil
Perfumary_Note - Base
Perfume family - Burseraceae
Good for blend with - Lemon, Myrrh, and other citrus and resins
**********
Oil name - Palmarosa Essential Oil
Perfumary_Note - Middle - Top
Perfume family - Poaceae
Good for blend with - Lavender, and other florals
**********
Oil name - Camphor- White Essential Oil
Perfumary_Note - Base
Perfume family - Lauraceae
Good for blend with - Eucalyptus, and other camphoraceous aromas
**********
Oil name - Pine- Scots Essential Oil
Perfumary_Note - Top
Perfume family - Pinaceae
Good for blend with - Myrrh, and other resins
**********
Oil name - Fir- Balsam Essential Oil
Perfumary_Note - Top - Middle
Perfume family - Pinaceae
Good for blend with - Myrrh, and other resins
**********
```

Evaluation:

After getting the results, I went through every algorithm and see its performance in terms of searching for best match. This is done manually where I go over the description of recommended fragrance ingredients and see is it related or similar to the query requested? Then, I will do scoring and measure accuracy based on the performance of each algorithm on multiple runs. The accuracy are as follows:

1. **Approximate Nearest Neighbor** (vector encoding using LSH) = 5.2 correctly matched on average out of 7 = **74.3 %**
2. **Approximate Nearest Neighbor** (vector encoding using trees) = 6.4 correctly matched on average out of 7 = **91.4 %**
3. **Okapi Best Match 25** (BM25) = 4.5 correctly matched on average out of 7 = **64.3 %**

After aggregating the results of multiple runs, I found that applying Approximate Nearest Neighbor with vector encoding using trees will have the best performance among other algorithms.

Conclusion:

Applying such techniques is powerful to tackle real life challenges that perfumers facing on daily basis. I found these searching algorithms useful in two ways:

- Helps me in finding an efficient solution to real life problem that I'm experiencing
- Saves time, effort and can be generalized to solve further issues

In fact, this is like a starting point of my research topic which is how to use the advancements of the fourth industrial revelation in perfumes industry.

References and Acknowledgement:

1. https://link.springer.com/referenceworkentry/10.1007/978-0-387-39940-9_921
2. https://towardsdatascience.com/comprehensive-guide-to-approximate-nearest-neighbors-algorithms-8b94f057d6b6
3. https://www.linkedin.com/in/abhijit-mahapatra807