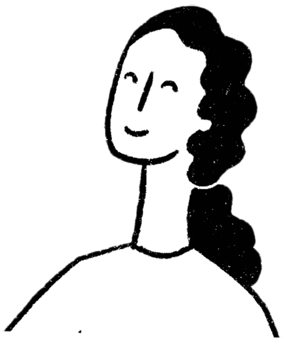
Vyno

The Digital Sommelier

A wine recommendation engine based on Machine Learning

Sebastiaan Dijkstra (12251267)
Dilyar Muslem Buzan (12645311)
Lowie de Beer (12674680)
Jacco Kortman (12603503)



Final report for
Tweedejaarsproject BSc KI

UNIVERSITEIT VAN AMSTERDAM
Netherlands
27 - 06 - 2021

Contents

1 Introduction

1.1 The Complex World of Wine

Wine is an incredibly complex world with an infinite number of varieties, regions, and styles. However, navigating this vast sea of choice can be a daunting task to the novice wine drinker. The choice between a bottle of red or white wine is just the tip of the iceberg. It is not just about the wine but also the people who make it and its culture. From vineyards in France to wineries in Napa Valley, many aspects of this drink are taken for granted. The people that study this complex world are called sommeliers.

1.2 Sommeliers

Sommeliers are the people that study wine and give advice on which wines to buy. They can be found in restaurants, fine dining establishments, or at a vineyard. They are not only experts at identifying wines but can tell each one apart by holding them up against the light or swirling it in front of their nose so that they may expose any imperfections before taking a sip themselves. They taste flavors and smell aromas that the average person does not even realize are there. When they then write about that wine, it is in language that is abstract and hard to understand. Unfortunately, this complexity leads to two problems: first, people stick to the same bottles of wine they have known their whole life, missing out on a whole variety of different flavor profiles they might like but do not know how to find. Second, this lack of experimentation and discovery hurts the wine merchants' conversions, as people do not know what to buy. Fortunately, Vyno has a solution to this problem.

1.3 Vyno

Vyno is a consumer insight platform for retailers, powered by Naomi, the artificial intelligence (AI) virtual sommelier. Naomi automates the wine profile of a merchant's portfolio and inputs them into a recommendation engine. Positioned as a chatbot on merchants' websites, Vyno enables intelligent wine recommendations for customers, collects consumer data, and produces actionable insight for retailers.

To test the market and gain initial traction, Vyno has currently built a rule-based recommendation engine. This engine works by predicting core attributes of wine profiles, such as body and acidity, refined with sommelier knowledge.

It works as follows: when a user visits a wine merchant that implemented Naomi, the chatbot shows up in the bottom right corner, telling the user what Naomi can do. Naomi then asks the user a couple of questions regarding specific tastes wrapped in a way that is easy to understand. Naomi will then recommend six wines, ranked from most to least, from the merchant's website.

The downside with the current system is that it is primarily rule-based, which means much time-consuming manual work is involved. For this reason, Vyno is looking to further develop the recommendations into the realm of AI. This project focuses on a small part of the overall vision of Vyno, namely that of live user interactions. Vyno wants a system that can utilize descriptions of wines to find similar products.

The final implementation they had in mind would consist of two phases. The first phase is to let the existing rule-based system narrow down the wine choice to 12 bottles. The second phase is a more modern AI solution that can reduce those 12 bottles to six wines ordered from most to least recommended. Vyno wanted this reduction to occur after the customer picks from eight words related to the taste of the wines. These words should be primary flavors like *lime* or *melon*, easily imagined by the average person reading them. This project will be implementing the second stage of this process.

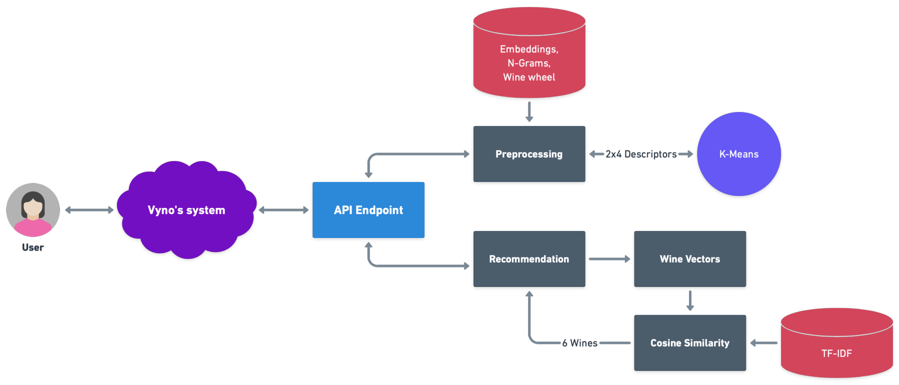
2 Method

This project consists of two different components. The first is creating the models, such as the embeddings, the second is making the actual recommendations. Figure 1 shows the final implementation. The system is going to work as follows:

1. The user interacts with the system of Vyno. The user answers a couple of questions that the chatbot asks. Vyno's system then generates 12 wines based on the answers to those questions and sends those wines and their descriptions to the recommendation system's API endpoint.
2. The system will then preprocess the wine descriptions using the models to extract the wines' key descriptors and are sent back to the user in two sets of four.
3. The user chooses two descriptors from both sets, which are sent back to the system.
4. The user descriptors are combined to create a single user preference vector and are then compared to the 12 wines on similarity.
5. The six most similar wines are returned to the user in order.

The next couple of pages will explain this process in more detail. Because the dataset in this project is all text-based and hidden semantic structures need to be found between the different wine descriptions, a clustering-based solution is the most logical. However, since clustering using text is impossible, the wine descriptions need to be converted to vector space, which requires much data for good results. Fortunately, Vyno possesses lots of wine data.

Figure 1: Overview of complete system.



2.1 Dataset

Vyno provided two datasets. The first one contained the country, description, name, province, region, subregion, grape, and vineyard for over 150,000 wines. The second dataset contained the same information, plus a column containing the wine title for an extra 130,000 wines. This column is a more detailed description of the wine name. Furthermore, the column order of both datasets was not equal, which resulted in some preprocessing before the datasets could be combined. Removing the duplicates from the merged dataset resulted in a dataset containing 169,000 wines.

2.2 Preprocessing

The first step in the embedding creation pipeline is the preprocessing of the wine descriptions. Taking a random wine sample from the dataset shows the following:

> "This tremendous 100% varietal wine hails from Oakville and was aged over three years in oak. Juicy red-cherry fruit and a compelling hint of caramel greet the palate, framed by elegant, fine tannins and a subtle minty tone in the background. Balanced and rewarding from start to finish, it has years ahead of it to develop further nuance. Enjoy 2022–2030."

Studying the descriptions shows that the text contains much information about the specific wine. In this particular example, keywords such as oak, fine tannins, and minty tones should accurately represent that wine. However, a couple of problems also appear. First of all, many stopwords have to be removed. Additionally, because wine descriptions have a distinct and creative language, many of the words in the description that look different describe the same taste.

After researching the literature and the web of previous work regarding natural language processing on wine descriptions, the following workflow from Schuring (2019) is adapted for the requirements of this project.

First, the descriptions are stripped of stop words and punctuation. Next, a stemmer is applied to get words back to their root form. For example, the word *tremendous* is stripped to *tremend* in the wine description example. Stemming words will keep the fluctuation of different words to a minimum, resulting in better embeddings later. In addition, many of the words in the description are combination words, such as *red cherry*. Because of these combination words, it is essential to look at the n-grams of every word to make sure the descriptions are accurately transferred to vector space later. N-grams are a contiguous sequence of *n* items from a given sample of text (*N-gram*, 2021). Because there are not many four-term words in the wine descriptions, only bi- and tri-grams will be used here. Because wine has a distinct language, a model is trained on all the wine descriptions to extract all the n-grams specific to wine language automatically. This trained n-gram model is saved for later use when preprocessing unseen descriptions. The wine description now looks as follows:

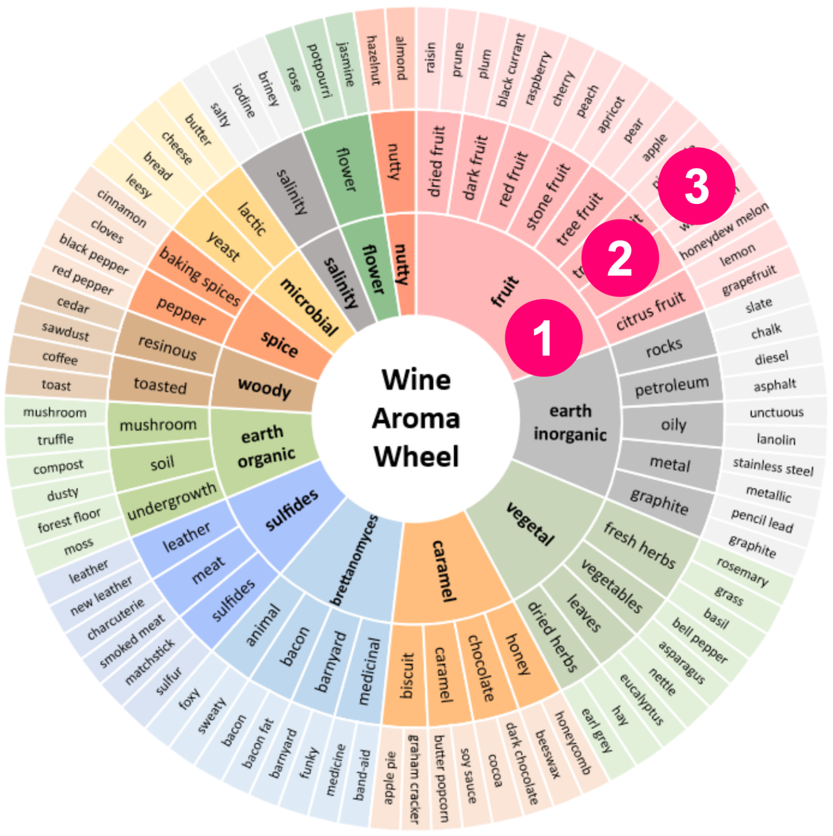
> "tremend, 100_variet, wine, hail, oakvill, age, three_year, oak, juici, redcherri, fruit, compel, hint, caramel, greet, palat, frame, eleg, fine, tannin, subtl, minti, tone, background, balanc, reward, start, finish, year, ahead, develop, nuanc, enjoy, 2022–2030"

The processed description shows that the stop words are correctly removed and that combination words such as red cherry and three years are combined into a single descriptor. Furthermore, words are brought back to their root form and are now ready for further processing.

2.3 Wine Aroma Wheel

As Schuring (2019) describes in his article, wine's language is very distinct and creative. Many of the words that seem different actually describe the same flavor and aroma. For example, *white peach*, *peachy*, and *peach nectarine* all describe the same flavor: *peach*. Bringing these words to their common denominator will significantly improve the embeddings. Fortunately, there has been much research around this area. Chen, Rhodes, Crawford, and Hambuchen (2014) have developed a computational wine wheel that maps words with the same meaning to a set of descriptors. Schuring (2019) developed this further by combining the research from Chen et al. (2014) with Wine Folly and UC Davis contributions. The result is a CSV file containing over 1000 descriptors and their mappings. Figure 2 shows an example of a wine wheel. This wheel works as follows: raw words in a wine description can be mapped to each of the three levels, depending on the level of abstraction needed. Figure **??** shows a concrete example with different words for the *apple* flavor. The raw descriptors *apple*, *apple cider*, and *apple flesh* can all be mapped to *apple* (level 3), *tree fruit* (level

Figure 2: Wine wheel (Schuring, 2019)

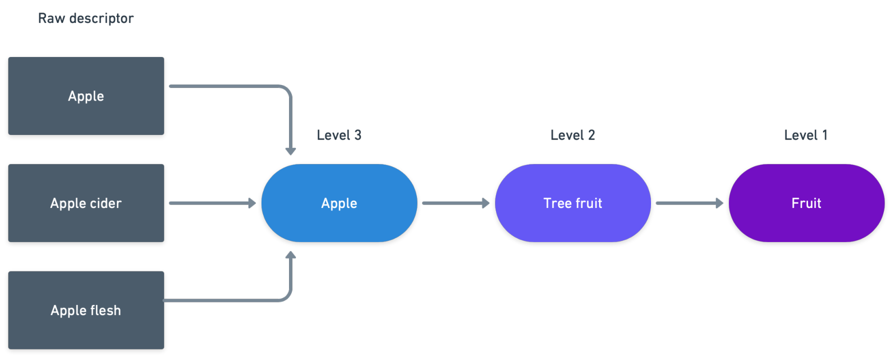


2), or *fruit* (level 1). Applying the descriptor map on sample descriptions yields the following result:

> "tremend, 100_variet, wine, hail, oakvill, age, three_year, oak, juicy, cherry, fruit, compel, hint, caramel, greet, palat, frame, elegant, fine, tannin, subtl, mint, tone, background, balanc, reward, start, finish, year, ahead, develop, nuanc, enjoy, 2022–2030"

Careful readers may notice that the *redcherri* is transformed to *cherry*, and rooted words like *eleg* are transformed to *elegant*. The descriptor map is applied to all the wine descriptions in the dataset. The next step is transforming the processed descriptions, such as the example above, to a numerical representation so that they can be clustered and compared to each other.

Figure 3: Example of descriptor mapping



2.4 Word2Vec

Word2Vec is used to retrieve the semantic meaning between different wine descriptors. Word2Vec works as follows: a two-layer neural network is trained on all the wine descriptions to reconstruct the contexts of the words. Word2Vec only needs one hidden layer and thus a two-layer neural network because the hidden layer weights are ultimately becoming the embeddings. One can check to make sure semantic meaning is learned by inputting a random word into the network and looking at similar returned words. For example, table 1 shows the output of the model when searching for the descriptor *lime*. As shown, these words are undoubtedly related to *lime*.

Descriptor	**Similarity**
citrus	0.759
lime_peel	0.745
lime_kiwi	0.696
nectarine	0.657
lemon	0.655

Table 1: Descriptors and similarity scores of the word *lime*.

2.5 TF-IDF

Because descriptors like *dry* and *fresh* are more common than descriptors such as *syrup* and *lemongrass*, the embeddings need a weighting to influence distinctive descriptors. These weights are created by calculating all descriptors' term frequency-inverse document frequency (TF-IDF) scores. The TF-IDF process works as follows: first, the amount of times a term is in a document (TF) is counted. Then, this is multiplied by the inverse of the number of documents the term shows up in (IDF).

$$\text{Weight} = tf * idf^{-1} \tag{1}$$

In the case of this project, the term frequency will almost always be 1, as a specific descriptor will very rarely occur more than once in a description. Here the inverse document frequency is the more exciting part as it describes how many times a particular term is used in descriptions. This means that a word like *dry* is given a much lower TF-IDF weight, as it is more commonly used to describe wines and therefore sets it less apart from the rest.

2.6 Making Recommendations

Now that all the models are ready, the only thing left is to create the actual recommendations. As mentioned previously, Vyno's rule-based system returns 12 wines and their description to this system. The goal of the recommendation system is to go from those 12 wine descriptions to eight unique descriptors that are easy to imagine and understand from the user's perspective. The user then picks four of those eight descriptors that most describe their taste. These descriptors are then compared to the 12 wine descriptions from which the six most similar are returned to the user in order.

The first step is figuring out how to go from 12 wine descriptions to eight keywords that both the user knows and represents all of the 12 wines. The wine descriptions are processed the same way as before, resulting in a list of descriptors for every wine. Not all of the descriptors are useful for the user, however. For instance, some wines are described as having *volcanic ash* as one of their tastes. This descriptor is not as relatable to most people as *apple*, for example. So the goal is only to show the user descriptors that relate to everyday flavors. Despite *volcanic ash* not being easy to understand as a taste, it is a helpful descriptor for comparing the wines from each other, so it is still used for making the comparisons. A custom list was created in collaboration with Vyno to filter out all the descriptors that are too abstract.

After applying the filter, all the descriptors are combined into a single set, thus removing duplicates. The filtering and processing already dramatically reduces the number of descriptors but are still not close to the number eight. K-means is used to get to eight descriptors.

K-means is a clustering algorithm that partitions n observations into k clusters in which each observation belongs to the cluster with the nearest cluster center (*K-means clustering*, 2021). What makes k-means great for this problem is that it is required to specify the number of clusters beforehand, whereas most other clustering algorithms decide how many clusters are required. In this case $k = 8$.

After the descriptors are clustered, the descriptor that lies closest to the cluster center is chosen as the returned descriptor, resulting in eight descriptors in total for eight clusters. The eight descriptors are then split into two sets of four and returned to the user. The user chooses two descriptors from both sets and is then returned to the recommendation system. The next step is comparing the four user-preferred descriptors to the 12 wine descriptions.

The user descriptors are transferred to vector space, multiplied by their respective TF-IDF weighting, and then summed to create a single user preference vector. The same happens for the 12 wine descriptions, transforming them into 12 wine vectors. The cosine similarity measure is used to compare the user- to the wine vectors.

$$\text{similarity}(U, W) = \cos(\theta) = \frac{\mathbf{U} \cdot \mathbf{W}}{\|\mathbf{U}\|\|\mathbf{W}\|} \tag{2}$$

Cosine similarity (2) is a measure of similarity between two different vectors by taking the cosine of the angle between them (*Cosine similarity*, 2021). It is calculated by taking the user vector $\mathbf{U}$, a wine vector $\mathbf{W}$, and taking their dot product divided by their magnitude. The results range from -1, meaning opposites, to 1, meaning identical.

After calculating the cosine similarity for every one of the 12 wines with the user vector, the six most similar wines are then returned to the user as recommendations in order.

2.7 API

Vyno's product specification asked for this model to be interactable through an API. The Flask framework is chosen for this project as it is easy to implement and fast (Grinberg, 2018).

There are two main request calls where Vyno will interact with the API. The first is a POST request to the server with a JSON file containing 12 wines and their descriptions. The API will run the previously described models and return two sets of four descriptors to the user. After the user makes its selection, the descriptors are sent back to the server as JSON objects to run the recommendation model. This model's output is then sent back to the user as a JSON object containing the six recommended wines.

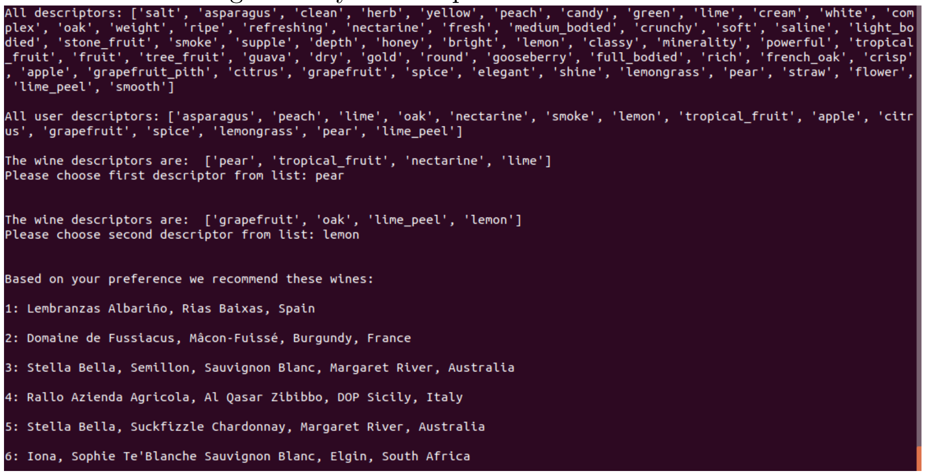
3 Results

This section compares three different inputs: 12 white wines, 12 red wines, and a mix of 12 red and white wines.

3.1 White Wines

Figure 4 shows the output of the system for the 12 white wines. For explanation's sake, all the descriptors are printed to the terminal to clarify the system. This information will not be returned to the user in production.

Figure 4: System output of the white wines.



```
All descriptors: ['salt', 'asparagus', 'clean', 'herb', 'yellow', 'peach', 'candy', 'green', 'lime', 'cream', 'white', 'com
plex', 'oak', 'weight', 'ripe', 'refreshing', 'nectarine', 'fresh', 'medium_bodied', 'crunchy', 'soft', 'saline', 'light_bo
died', 'stone_fruit', 'smoke', 'supple', 'depth', 'honey', 'bright', 'lemon', 'classy', 'minerality', 'powerful', 'tropical
_fruit', 'fruit', 'tree_fruit', 'guava', 'dry', 'gold', 'round', 'gooseberry', 'full_bodied', 'rich', 'french_oak', 'crisp'
, 'apple', 'grapefruit_pith', 'citrus', 'grapefruit', 'spice', 'elegant', 'shine', 'lemongrass', 'pear', 'straw', 'flower',
 'lime_peel', 'smooth']

All user descriptors: ['asparagus', 'peach', 'lime', 'oak', 'nectarine', 'smoke', 'lemon', 'tropical_fruit', 'apple', 'citr
us', 'grapefruit', 'spice', 'lemongrass', 'pear', 'lime_peel']

The wine descriptors are:  ['pear', 'tropical_fruit', 'nectarine', 'lime']
Please choose first descriptor from list: pear


The wine descriptors are:  ['grapefruit', 'oak', 'lime_peel', 'lemon']
Please choose second descriptor from list: lemon


Based on your preference we recommend these wines:

1: Lembranzas Albariño, Rias Baixas, Spain

2: Domaine de Fussiacus, Mâcon-Fuissé, Burgundy, France

3: Stella Bella, Semillon, Sauvignon Blanc, Margaret River, Australia

4: Rallo Azienda Agricola, Al Qasar Zibibbo, DOP Sicily, Italy

5: Stella Bella, Suckfizzle Chardonnay, Margaret River, Australia

6: Iona, Sophie Te'Blanche Sauvignon Blanc, Elgin, South Africa
```

When comparing the descriptions with the extracted keywords, the resulting user descriptors make much sense. For example, descriptors such as *grapefruit*, *tropical fruit*, and *lemon* are easy to imagine. Furthermore, the two sets of four descriptors returned to the user are distinct and easy to understand.

In this example, the user chooses *pear* and *lemon*, resulting in six wine recommendations. Looking at the description of the most recommend wine:

> "Lemon yellow in colour with hints of gold. Citrus fruit hits the nose followed by apple, pear and dried candied fruits with a mineral note. . . ".

This description seems to correctly fit the chosen descriptors, seeing that the chosen descriptors *pear* and *lemon*, in the form of citrus, are found in the description. Considering the description of the wine at second place:

> "Fragrant and fresh yet well rounded with subtle peachy fruit and touches of mineral character."

On the first impression, the second wine does not seem to match the descriptors as well as the first description. It is hard to confirm this wine is correctly placed considering the algorithm uses all flavors in the description for its vector and might make the right profile. However, it is impossible to tell if this profile matches without actually tasting all the wines. Considering these facts, it seems complicated to confirm that the Domaine de Fussiacus is similar to the Lembranzas Albariño, but maybe the third place will provide more clarity. Looking at the description of the wine at the third spot in the list:

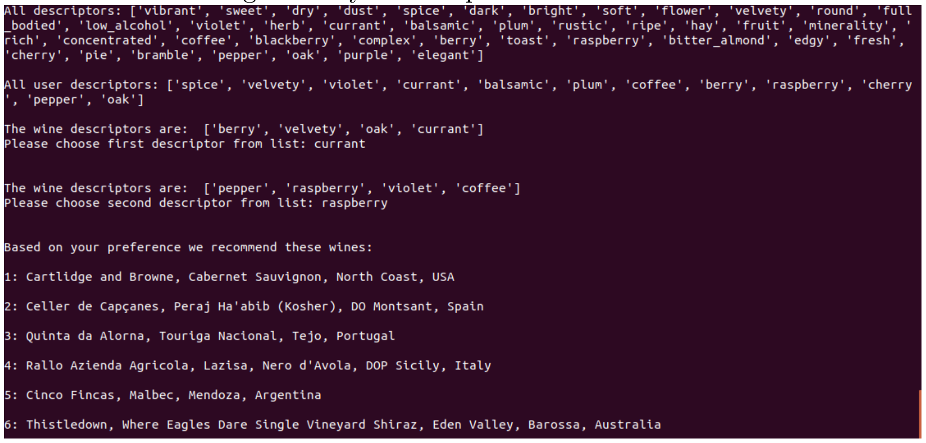
> "A charming combination of citrus blossom, lemon, lime and guava with a hint of musk, candied apple and floral aromas."

Comparing this wine with the other two, it seems that it is at the right spot in the list, as this wine is dryer and more floral than the higher-ranked wines. Because Sauvignon Blanc is usually a dry white wine and contains sour and floral elements, the system used the lemon descriptor to place this third. Seen the fact that the sauvignon blanc and albariño are very acidic and mineral wines they fit the picture, the chardonnay in second place has a round and minerally taste as stated in its description so that it might fit the profile, but that still is tough to tell without actually tasting the wine.

3.2 Red Wines

Figure 5 shows the output of the system for the 12 red wines. Descriptors such as *spice*, *velvety*, *oak*, and *blackberry* are common characteristics of red wines. Furthermore, the descriptors shown to the user are all easy to imagine for the average person.

Figure 5: System output of the red wines.



```
All descriptors: ['vibrant', 'sweet', 'dry', 'dust', 'spice', 'dark', 'bright', 'soft', 'flower', 'velvety', 'round', 'full
_bodied', 'low_alcohol', 'violet', 'herb', 'currant', 'balsamic', 'plum', 'rustic', 'ripe', 'hay', 'fruit', 'minerality', '
rich', 'concentrated', 'coffee', 'blackberry', 'complex', 'berry', 'toast', 'raspberry', 'bitter_almond', 'edgy', 'fresh',
'cherry', 'pie', 'bramble', 'pepper', 'oak', 'purple', 'elegant']

All user descriptors: ['spice', 'velvety', 'violet', 'currant', 'balsamic', 'plum', 'coffee', 'berry', 'raspberry', 'cherry
', 'pepper', 'oak']

The wine descriptors are:  ['berry', 'velvety', 'oak', 'currant']
Please choose first descriptor from list: currant


The wine descriptors are:  ['pepper', 'raspberry', 'violet', 'coffee']
Please choose second descriptor from list: raspberry


Based on your preference we recommend these wines:

1: Cartlidge and Browne, Cabernet Sauvignon, North Coast, USA

2: Celler de Capçanes, Peraj Ha'abib (Kosher), DO Montsant, Spain

3: Quinta da Alorna, Touriga Nacional, Tejo, Portugal

4: Rallo Azienda Agricola, Lazisa, Nero d'Avola, DOP Sicily, Italy

5: Cinco Fincas, Malbec, Mendoza, Argentina

6: Thistledown, Where Eagles Dare Single Vineyard Shiraz, Eden Valley, Barossa, Australia
```

In this example, the user chooses the following descriptors: *currant* and *raspberry*, both forest fruits. Looking at the description of the most recommend wine:

> "This Cabernet starts out with rich, delicious red fruits on the nose that flows into cherry, watermelon, candied plum and crème brûlée."

The first placed Cabernet Sauvignon description involves much red fruit, so it should be compared to the second and third place to see if it deserves the number one spot. The second-place descriptions read as follows:

> "Dark, full red yet vibrant colour. Bright aromas of cherry, plums, currant and toasty, coffee, mocha nuances."

The description describes the wine as having a bright cherry aroma, so forest fruits are present but less than in the Cabernet Sauvignon. Seeing that the aroma of a wine is a big part of the taste, it is not surprising that this wine got the number two spot. Looking at the third spot:

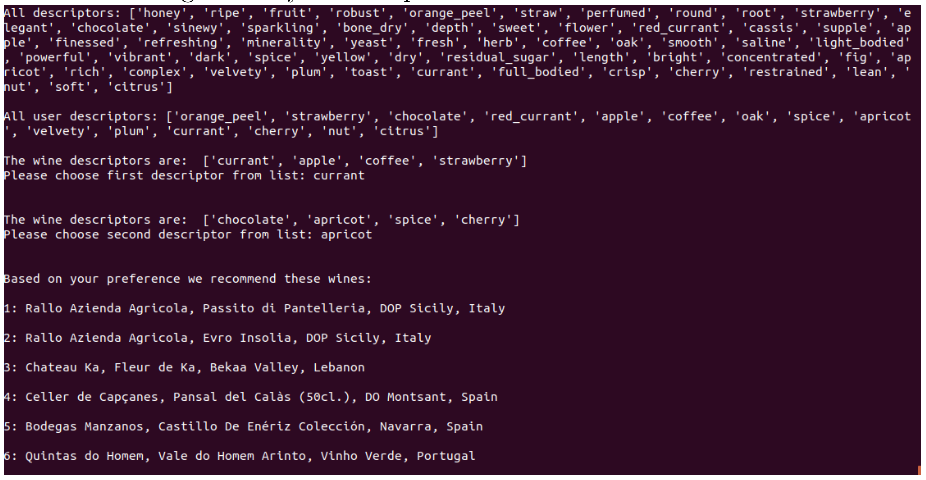
> "This purple coloured wine has a floral aroma with strong notes of violets and red and black fruits, like raspberry and blackberry."

This wine has forest fruits as strong notes, which is less present than the aroma. So the ranking of red wines can be considered accurate as well.

3.3 Mixed Wines

Since customers of Vyno can indicate they prefer both red and white wine, it is essential to have a look at the combined results as well. Figure 6 shows these results.

Figure 6: System output of the red and white wines.



```
All descriptors: ['honey', 'ripe', 'fruit', 'robust', 'orange_peel', 'straw', 'perfumed', 'round', 'root', 'strawberry', 'e
legant', 'chocolate', 'sinewy', 'sparkling', 'bone_dry', 'depth', 'sweet', 'flower', 'red_currant', 'cassis', 'supple', 'ap
ple', 'finessed', 'refreshing', 'minerality', 'yeast', 'fresh', 'herb', 'coffee', 'oak', 'smooth', 'saline', 'light_bodied'
, 'powerful', 'vibrant', 'dark', 'spice', 'yellow', 'dry', 'residual_sugar', 'length', 'bright', 'concentrated', 'fig', 'ap
ricot', 'rich', 'complex', 'velvety', 'plum', 'toast', 'currant', 'full_bodied', 'crisp', 'cherry', 'restrained', 'lean', '
nut', 'soft', 'citrus']

All user descriptors: ['orange_peel', 'strawberry', 'chocolate', 'red_currant', 'apple', 'coffee', 'oak', 'spice', 'apricot
', 'velvety', 'plum', 'currant', 'cherry', 'nut', 'citrus']

The wine descriptors are:  ['currant', 'apple', 'coffee', 'strawberry']
Please choose first descriptor from list: currant


The wine descriptors are:  ['chocolate', 'apricot', 'spice', 'cherry']
Please choose second descriptor from list: apricot


Based on your preference we recommend these wines:

1: Rallo Azienda Agricola, Passito di Pantelleria, DOP Sicily, Italy

2: Rallo Azienda Agricola, Evro Insolia, DOP Sicily, Italy

3: Chateau Ka, Fleur de Ka, Bekaa Valley, Lebanon

4: Celler de Capçanes, Pansal del Calàs (50cl.), DO Montsant, Spain

5: Bodegas Manzanos, Castillo De Enériz Colección, Navarra, Spain

6: Quintas do Homem, Vale do Homem Arinto, Vinho Verde, Portugal
```

The chosen descriptor *currant* is mainly used for red wines and the chosen descriptor *apricot* primarily for white wines. This combination will make it particularly interesting to see how the ranking will be calculated and which descriptor ranks higher. Looking at the description of the most recommend wine:

> "A light amber colour with intense yet sweet, elegant and well-balanced with notes of orange peel, apricots, dried figs and honey with impressive acidity."

This description describes the wine as having notes of apricot, which is not significantly dominant, but the wine does include apricot. Now it is essential to see if the number two wine has either currant or apricot as a descriptor but less presently:

> "A straw yellow colour wine, with fresh citrussy notes and crisp apple. Persistent harmanious and mineral on the finish."

Unfortunately, this wine does neither include apricot or currant, making it hard to see why the recommendation engine has placed this wine second. It is plausible that it has calculated currant as a dry and sour fruit that, collectively with the apricot's vector, makes this sour and dry wine suitable. The third-place must make less sense than the second place to see if the engine is entirely accurate:

> "Bright red fruits jump out of the glass along with a hint of oak. The palate is rich, smooth and supple with bright cherry and redcurrant fruit."

This wine contains the word redcurrant and seems to fit the forest fruits taste very well. However, the apricot is missing. The missing apricot is probably why this wine ranks third: it fits half of the chosen descriptors, so it is a pretty good recommendation but not fitting for both descriptors.

4 Discussion

In conclusion, it is safe to say the algorithm operates rather accurately. When analyzing the results, it is hard to determine whether it works perfectly. The difficulty in this is because the taste of wine can be very subjective, and it is impossible to check if wines are similar enough based solely on the description. Luckily this project was mainly about making a product for Vyno that was easy to use and would contribute to their virtual sommelier. Since day one, there has been intensive contact between the company and the engineers, which has contributed to a streamlined end product that should tick off many boxes from Vyno's expectations.

There is always room for improvement, so here a few points of improvement will be discussed. The first point is that some descriptors in the wine wheel can be viewed as too vague or simply unsuitable to present to a customer. Furthermore, Vyno might decide that a descriptor is missing, but this is easy to deal with since the wine wheel is a CSV that can be altered. Anyone using the recommendation engine can open the CSV file and add or delete words without any problem since it does not alter the code.

Another point of improvement can be found when a customer selects both white and red wines as the desired output. The descriptors shown are mixed with white and red wines and do not necessarily feel natural in combination. When reading *cherry* and *apricot* next to each other, it is hard to imagine a good wine using both these descriptors. The optimal solution would be to use only red wine descriptors for the first four words and solely white wine descriptors for the second four words.

References

Chen, B., Rhodes, C., Crawford, A., & Hambuchen, L. (2014). Wineinformatics: applying data mining on wine sensory reviews processed by the computational wine wheel. In *2014 ieee international conference on data mining workshop* (pp. 142–149).

Cosine similarity. (2021, Apr). Wikimedia Foundation. Retrieved from `https://en.wikipedia.org/wiki/Cosine`$_s$*imilarity*

Grinberg, M. (2018). *Flask web development: developing web applications with python.* " O'Reilly Media, Inc.".

K-means clustering. (2021, Jun). Wikimedia Foundation. Retrieved from `https://en.wikipedia.org/wiki/K-means`$_c$*lustering*

N-gram. (2021, Apr). Wikimedia Foundation. Retrieved from `https://en.wikipedia.org/wiki/N-gram`

Schuring, R. (2019, Dec). *Robosomm chapter 3: Wine embeddings and a wine recommender.* Towards Data Science. Retrieved from `https://towardsdatascience.com/robosomm-chapter-3-wine-embeddings-and-a-wine-recommende`