Introduction

The Dungeons and Dragons Spells (2024) dataset, which details 315 spells from the 2024 edition of the Dungeons and Dragons Free Rules. Each spell is characterized by 27 attributes, including its name, level (0–9), school of magic, casting components (verbal, somatic, material), range, duration, and the classes that can cast it (e.g., Cleric, Wizard). The dataset originates from the official rules published by Wizards of the Coast, offering a structured yet creative lens into the game's magical mechanics. Understanding these spells is significant for players and designers alike, as they reveal patterns in game balance and class identity. However, challenges like inconsistent range formats (e.g., "30 feet" vs. "touch") and subjective text descriptions require careful preprocessing to ensure meaningful analysis.

We chose the Dungeons and Dragons spells dataset because it combines creativity with structured attributes, making it ideal for data visualization. The dataset's rich details—such as spell levels, schools of magic, casting times, and components—provide diverse variables to analyze and present. Additionally, the classification of spells by character classes adds an intriguing layer of complexity, appealing to both our interest in fantasy and our aim to demonstrate effective visualization techniques for categorical and numerical data.

Question 1: How do common words used in spell names and descriptions vary in accordance with the classes that can use the spell?

<u>Introduction</u>

This question aims to discover how DND class characteristics affect language usage. Diverse classes are one of the key characteristics of DND, with each of them having its own unique capabilities, play style, and character archetypes. Over time, the DND community developed various stereotypes about each class: for example, bards are rumored to always be looking for love, wizards are depicted as a fantasy variant of nerds, druids are said to hate civilization, etc.. The purpose of this question is to analyze the words used in the names and descriptions of spells open to each of the classes to see if these stereotypes have basis in the vocabulary used when officially defining the class. As DND players, we were very curious to see just how much the common cliches are grounded in the rulebook.

To discover the answer, we used the 'name' and 'description' columns in the dataset, as well as the 8 columns corresponding to each of 8 classes that can cast spells: bard, cleric, druid, paladin, ranger, sorcerer, warlock, and wizard. The 'name' column contains a phrase (typically 1-3 words) that serves as the name of a spell; the 'description' column contains a few sentences describing its in-game mechanics application, and the class columns contain boolean markers of whether a certain class can use a certain spell.

<u>Approach</u>

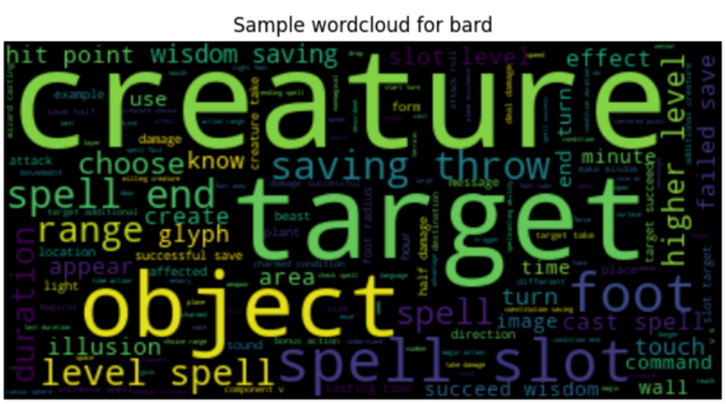
Since this is a question of discovering the most frequent words in text documents, we chose to answer it with a wordcloud. To better highlight the differences between classes, as well as to give the viewer the opportunity to switch between names and descriptions, we made the visualization interactive, with 2 corresponding drop-down menus. The class menu allows for multiple selection to assist in identifying themes that classes might have in common.

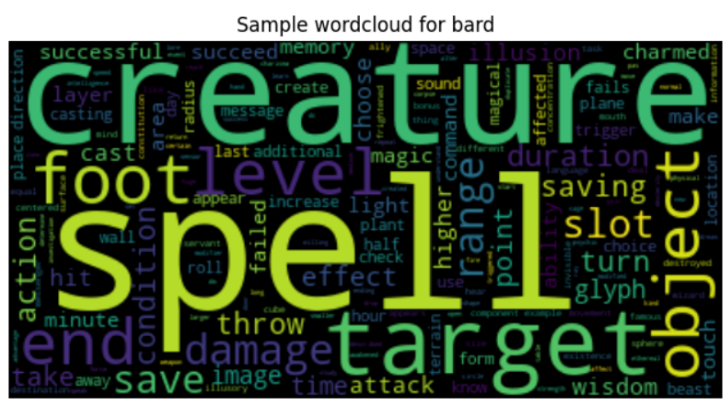
An interesting feature of our approach is the text processing. Because spell descriptions are highly specific to the game to begin with, word frequency and TF-IDF yielded general words referencing the in-game spell mechanics as the most common ones. Therefore, to focus on class-specific words, it was decided to use PMI (pointwise mutual information), a measure of how strongly a word is associated with a certain class compared to random distribution, to score the words by.

<u>Analysis</u>

We treated all names/descriptions available to a class as one document. This is the code that generates them from the processed text columns. The processing itself is very standard and is therefore omitted from this report.

```
# get unified lists of words for each class

classes = ['bard', 'cleric', 'druid', 'paladin', 'ranger', 'sorcerer', 'warlock', 'wizard']

names = {}
descriptions = {}

for cl in classes:
    names[cl] = [x for xx in df[df[cl] == True]['name_p'] for x in xx]
    descriptions[cl] = [x for xx in df[df[cl] == True]['description_p'] for x in xx]
```





Sample wordclouds for the bard class obtained with word frequency and TF-IDF on spell descriptions

This is the main code section of the PMI computation

```python
# compute PMI scores

pmi_scores_names = {}
pmi_scores_descriptions = {}

for cl in classes:
    total_names_class = sum(name_counts[cl].values())
    total_descriptions_class = sum(description_counts[cl].values())

    pmi_scores_names[cl] = {}

    for word, count in name_counts[cl].items():
        p_w_given_c = count / total_names_class  # P(w | c)
        p_w = name_counts_total[word] / total_names  # P(w)

        if p_w_given_c > 0 and p_w > 0:
            pmi = np.log(p_w_given_c / p_w)
            pmi_scores_names[cl][word] = pmi

    pmi_scores_descriptions[cl] = {}

    for word, count in description_counts[cl].items():
        p_w_given_c = count / total_descriptions_class  # P(w | c)
        p_w = description_counts_total[word] / total_descriptions  # P(w)

        if p_w_given_c > 0 and p_w > 0:
            pmi = np.log(p_w_given_c / p_w)
            pmi_scores_descriptions[cl][word] = pmi
```

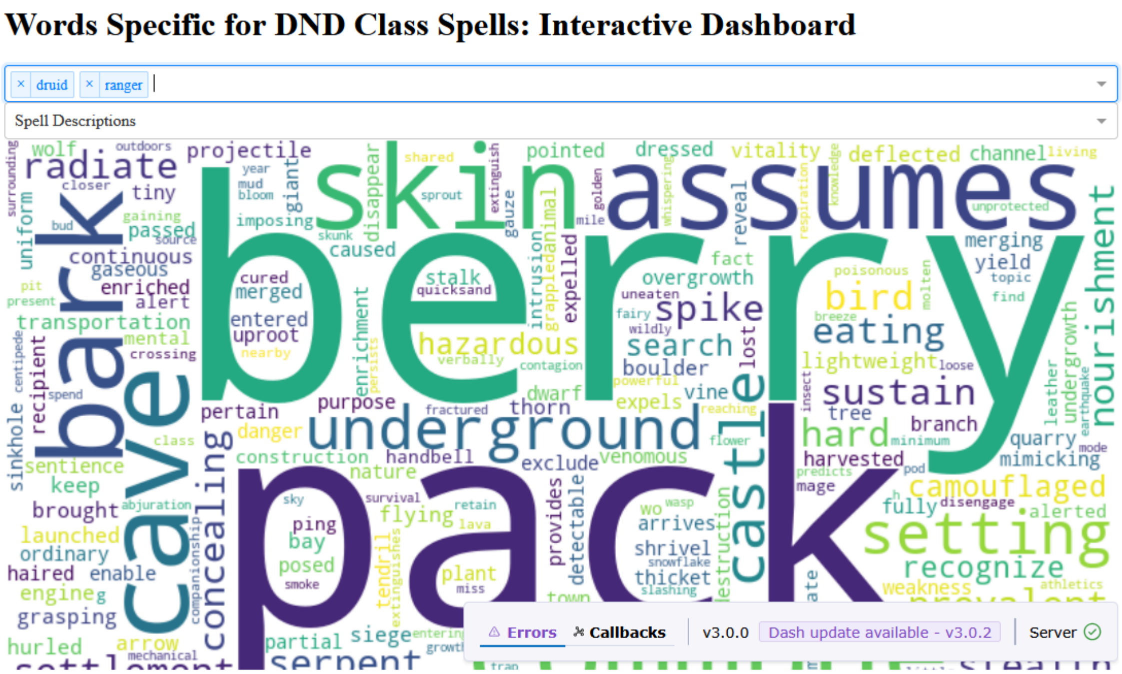
This is the code that generates the final interactive visualization. Because interactive wordclouds are not built in in Dash, it saves the visualization as an image and displays it every time a change is made in the menu.

```python
# initialize Dash app
app = dash.Dash(__name__)

# layout
app.layout = html.Div([
    html.H1('Words Specific for DND Class Spells: Interactive Dashboard'),

    # dropdown for class selection
    dcc.Dropdown(
        id='class-filter',
        options=[{'label': cl, 'value': cl} for cl in classes],
        value=['wizard'],
        multi=True,
        clearable=False,
    ),

    # dropdown to switch between names and descriptions
    dcc.Dropdown(
        id='column-filter',
        options=[
            {'label': 'Spell Names', 'value': 'names'},
            {'label': 'Spell Descriptions', 'value': 'descriptions'}
        ],
        value='names',
        clearable=False,
    ),

    # wordcloud output (image)
    html.Img(id='wordcloud', style={'width': '100%', 'height': 'auto'})
])
```

```python
# update chart based on the selection
@app.callback(
    Output('wordcloud', 'src'),
    [Input('class-filter', 'value'),
     Input('column-filter', 'value')]
)
def update_wordcloud(selected_classes, selected_column):

    # select the correct dataset
    if selected_column == 'names':
        data = names_pmi.copy()
    elif selected_column == 'descriptions':
        data = descriptions_pmi.copy()

    # filter by selected classes
    if selected_classes:
        data = data[selected_classes].sum(axis=1)

    # generate wordcloud
    wc = WordCloud(width=800, height=400, background_color='white').generate_from_frequencies(data.to_dict())

    # convert wordcloud to image
    img = io.BytesIO()
    wc.to_image().save(img, format='PNG')
    img.seek(0)

    # encode image to display in Dash
    encoded_img = base64.b64encode(img.getvalue()).decode()
    return f'data:image/png;base64,{encoded_img}'


# run
if __name__ == '__main__':
    app.run(port=8050, debug=True, use_reloader=False)
```

A few examples of the final visualization:

Words Specific for DND Class Spells: Interactive Dashboard



Discussion (1-3 paragraphs):
While our analysis revealed that the vocabulary used in the descriptions of spells for different classes has more inter-class similarities than differences, these words are the ones describing the in-game mechanics of spell (creature, target, spell, object, etc.). Once a representation favoring class-specific words was chosen, the individual trends of each class became apparent. Overall, it seems that the common stereotypes are indeed based on the rulebook, but were exaggerated by the player community for some classes more than for the others: for example, the wordclouds for the druid class mostly consist of nature-related words, but the ones for the bard class lean more towards their mastery of performance, and less towards the common personality cliches.

Question 2: What influence do the level of the spell and the presence of verbal, somatic, and material components have on its range?

Introduction
This question explores how a spell's level and its required components—verbal (V), somatic (S), and material (M)—influence its range in the Dungeons and Dragons Spells (2024) dataset. We focus on the "level" column (ranging from 0 to 9), the "range" column (e.g., "touch," "30 feet," "sight"), and the binary indicators for components ("verbal," "somatic," "material"). Understanding these relationships is intriguing because range often determines a spell's tactical utility in gameplay, and we hypothesize that higher-level spells or specific component combinations might enable greater ranges, reflecting their complexity or power.

Approach

To address this question, we use three distinct visualizations to uncover patterns between spell level, components, and range. First, a faceted scatter plot displays range versus spell level, with separate facets for spells with one, two, or three components, using color to differentiate component counts. This plot is ideal for observing how range distributes across levels and component counts, with faceting enhancing readability for the dense dataset. Second, a line trend plot shows average range by component count (0–3) across levels, chosen to highlight trends in range as component complexity increases. Third, an interactive heatmap represents average range with spell level on rows and component combinations (e.g., "V," "V+S," "V+S+M") on columns, allowing hover details for specific spells. The heatmap effectively summarizes trends across multiple dimensions, leveraging color mapping to emphasize range variations.

Minor preprocess:

```
df = pd.read_csv('spells.csv')

df['num_components'] = (df['verbal_component'].astype(int) +
                        df['somatic_component'].astype(int) +
                        df['material_component'].astype(int))

def convert_range(r):
    if pd.isna(r):
        return np.nan
    r = r.lower().strip()
    if 'feet' in r:
        return float(r.replace('feet', '').strip())
    elif r == 'self' or r == 'touch':
        return 0  # self/touch = 0 feet
    elif r == 'sight' or 'mile' in r:
        return 1000  # arbitrary val for sight/miles
    else:
        return np.nan # others as nan

df['range_numeric'] = df['range'].apply(convert_range)

df_clean = df.dropna(subset=['range_numeric'])
```

<u>Analysis</u>

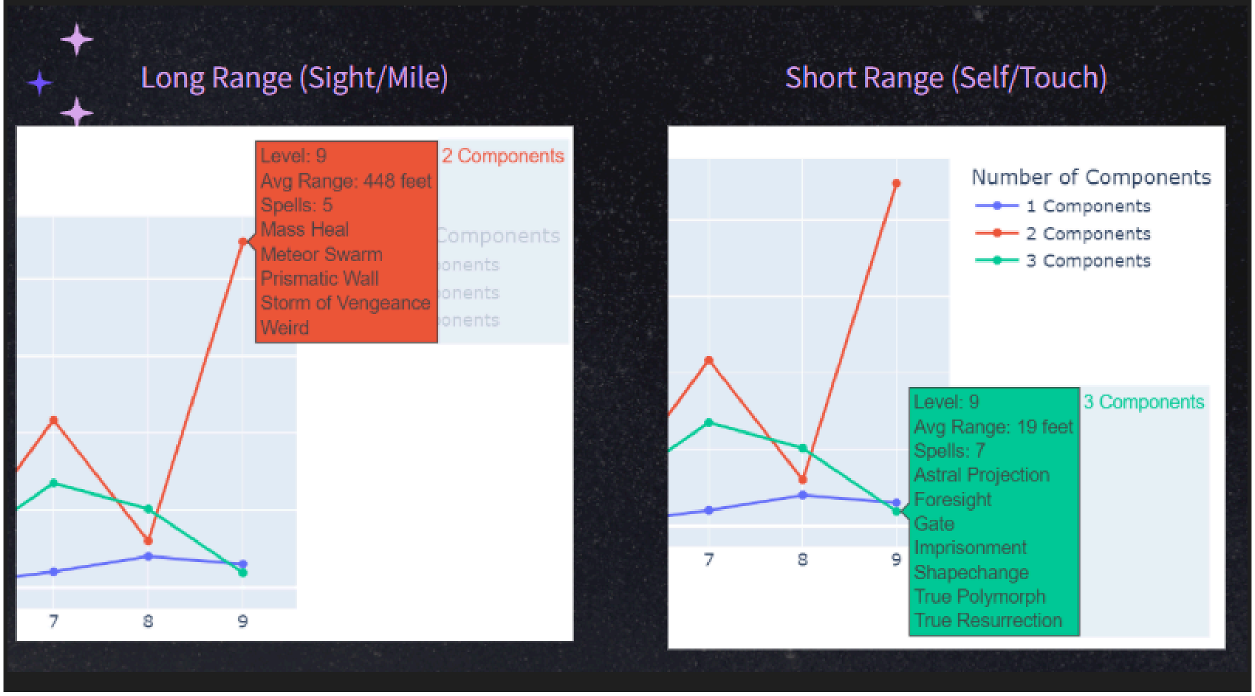
The faceted scatter plot revealed a well-distributed spread of ranges across spell levels, with most spells clustering between 0 and 200 feet. Notably, spells with only one component (e.g., "V" alone) never reach long ranges like "sight" or "mile," and there’s a significant gap beyond 150 feet where few spells exist. Zooming into the 0–200 feet range clarified these patterns, as outliers like "sight" skewed the full view.

The line trend plot highlighted that spells with two components (e.g., "V+S") are more likely to achieve longer ranges, such as "sight" or "mile," while spells with three components ("V+S+M") tend toward shorter ranges, often below 100 feet. This suggests a non-linear relationship between component count and range.

Interactive line chart:

```
import plotly.graph_objects as go

trend_data = df_clean.groupby(['level', 'num_components']).agg(
    range_numeric=('range_numeric', 'mean'),
    spell_count=('name', 'count'),
    spell_names=('name', lambda x: '<br>'.join(x))
).reset_index()


fig = go.Figure()
for num in range(1, 4):
    subset = trend_data[trend_data['num_components'] == num]
    if not subset.empty:
        fig.add_trace(go.Scatter(
            x=subset['level'], y=subset['range_numeric'],
mode='lines+markers',
            name=f'{num} Components',
            hovertemplate=(
                'Level: %{x}<br>'
                'Avg Range: %{y:.0f} feet<br>'
                'Spells: %{customdata[0]}<br>'
                '%{customdata[1]}'
            ),
            customdata=subset[['spell_count', 'spell_names']].values
        ))

fig.update_layout(
    title='Average Range by Level and Number of Components',
    xaxis_title='Spell Level', yaxis_title='Average Range (feet)',
    legend_title='Number of Components',
    xaxis=dict(tickmode='linear', tick0=0, dtick=1)
)
```

```
fig.show()
```



The interactive heatmap further clarified these trends: spells with material components ("M") are strongly associated with short ranges like "self" or "touch," regardless of level. In contrast, spells with only verbal and somatic components ("V+S") show ranges that scale with level, reaching higher averages at upper levels (e.g., 7–9). Hovering revealed specific examples, such as high-level "V+S" spells with "sight" range versus "V+S+M" spells limited to "touch."

Interactive heatmap:

```
import plotly.express as px
import pandas as pd

# hover info
df_clean['spell_info'] = df_clean['name'] + ': ' +
df_clean['description'].str.slice(0, 50) + '...'
pivot_range = pd.pivot_table(df_clean, values='range_numeric',
                             index='level', columns='component_combo',
                             aggfunc='mean')
pivot_spells = pd.pivot_table(df_clean, values='spell_info',
                              index='level', columns='component_combo',
                              aggfunc=lambda x: '<br>'.join(x))

fig = px.imshow(pivot_range,
                labels=dict(x='Component Combination', y='Spell Level',
color='Avg Range (feet)'),
```

```
                  title='Average Spell Range by Level and Component Combination')
fig.update_traces(hovertemplate='Level: %{y}<br>Combo: %{x}<br>Avg Range: %{z:.0f} feet<br>Spells:<br>%{customdata}',
                    customdata=pivot_spells.values)
fig.show()
```

<u>Discussion</u>

The analysis reveals distinct trends in how spell level and components influence range. Higher-level spells do not universally have longer ranges; instead, range depends heavily on component combinations. Spells with two components, particularly "V+S," enable longer ranges like "sight" or "mile," possibly reflecting a balance of complexity and flexibility in casting. Conversely, the addition of material components ("M") often restricts range to "self" or "touch," perhaps due to their role in spells targeting the caster or immediate vicinity, such as buffs or transformations. The gap beyond 150 feet suggests a design choice in Dungeons and Dragons to limit mid-to-long-range options, focusing most spells on close combat or utility. These patterns align with gameplay mechanics where material components might ground a spell's effect locally, while verbal and somatic pairings allow projection over distance. Speculatively, this could reflect lore-based constraints or balance considerations in the 2024 ruleset.

Conclusion:

Our work of the Dungeons and Dragons Spells (2024) dataset analysis provided valuable insights into the interplay between spell attributes and their implications for gameplay and design. For Question 1, examining spell names and descriptions across classes revealed that spell names often align with class stereotypes (e.g., "healing" for Clerics, "illusion" for Wizards), while descriptions uncover common mechanics and language that match class characteristics, reflecting thematic roles in the game. For Question 2, we found that spell range is influenced more by component combinations than level alone: spells with verbal and somatic components ("V+S") scale with level to achieve longer ranges like "sight," whereas material components

("M") anchor spells to shorter ranges like "self" or "touch," with a notable gap beyond 150 feet suggesting intentional design limits.

These findings highlight how spell design balances creativity and mechanics, offering players tactical variety while maintaining class identity and range constraints. The visualizations—word clouds, scatter plots, line trends, and heatmaps—effectively illuminated these patterns, blending textual and numerical insights. For future work, broadening the dataset to include extended rulebook details (e.g., spellcasting conditions or lore) could deepen our understanding of these trends. Further exploration of the current dataset, such as analyzing casting time or school of magic, might also reveal additional layers of complexity in spell mechanics. Ultimately, this project underscores the potential of data visualization to decode the intricate systems underlying Dungeons and Dragons, paving the way for both strategic player decisions and informed game design refinements