In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sn
import matplotlib.pyplot as plt
import re
import nltk
from collections import Counter
from sklearn.feature_extraction.text import TfidfVectorizer,CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB,MultinomialNB
from sklearn.svm import SVC
from sklearn import metrics
```

https://www.kaggle.com/code/pythonafroz/roberta-book-summary-predictiono

Join Our Telegram Channel to Learn AI & ML: https://t.me/AIMLDeepThaught

In [2]:
```
books = pd.read_csv('/kaggle/input/bookdata/BooksDataSet.csv')
books = books.drop(['Unnamed: 0'], axis=1)
books.head()
```

Out[2]:

	book_id	book_name	genre	summary
0	3248537	Drowned Wednesday	Fantasy	Drowned Wednesday is the first Trustee among ...
1	27796919	The Lost Hero	Fantasy	As the book opens, Jason awakens on a school ...
2	3910776	The Eyes of the Overworld	Fantasy	Cugel is easily persuaded by the merchant Fia...
3	5969644	Magic's Promise	Fantasy	The book opens with Herald-Mage Vanyel return...
4	3173445	Taran Wanderer	Fantasy	Taran and Gurgi have returned to Caer Dallben...

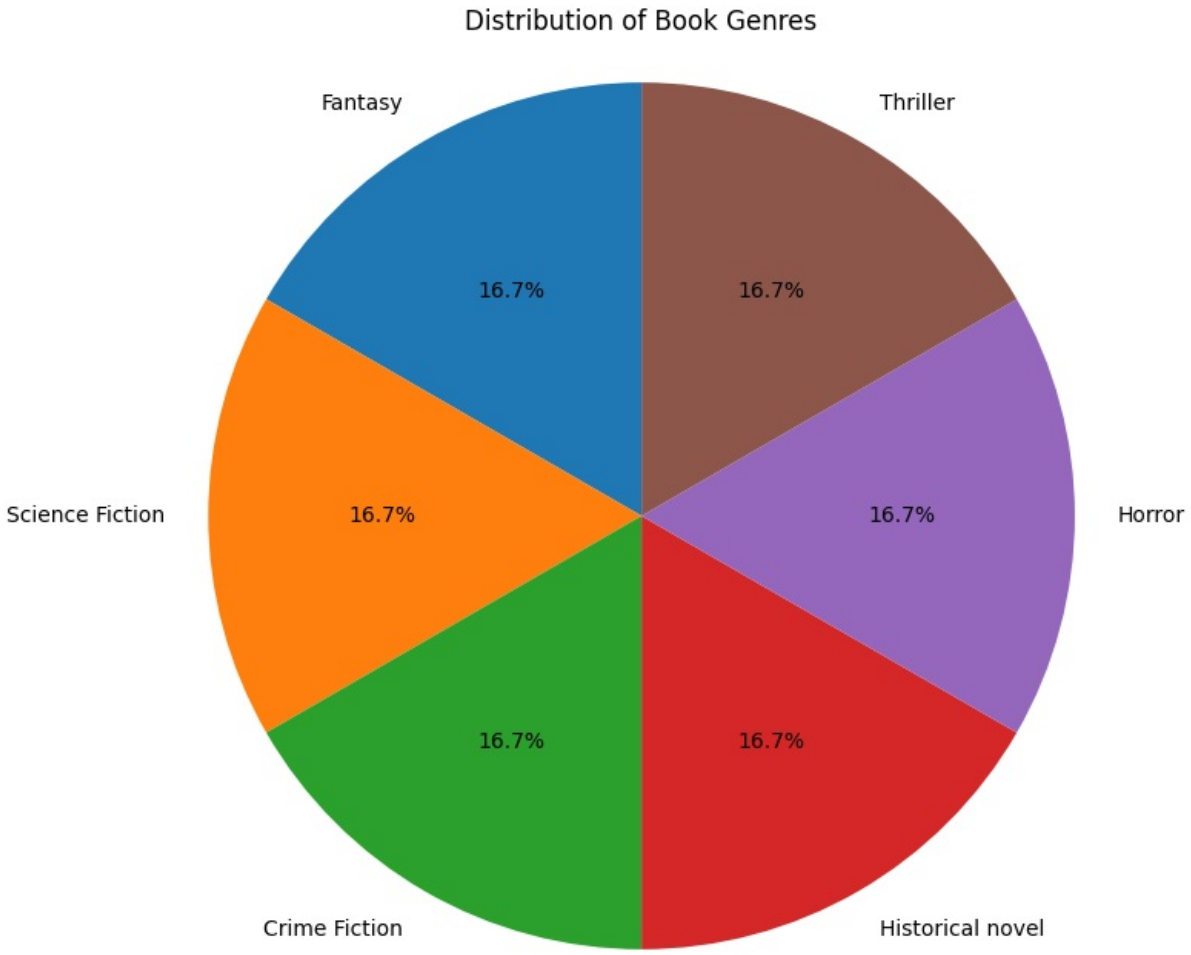
In [3]:
```
books['genre'].value_counts()
```

Out[3]:
```
genre
Fantasy             500
Science Fiction     500
Crime Fiction       500
Historical novel    500
Horror              500
Thriller            500
Name: count, dtype: int64
```

In [4]:
```
books.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   book_id    3000 non-null   int64 
 1   book_name  3000 non-null   object
 2   genre      3000 non-null   object
 3   summary    3000 non-null   object
dtypes: int64(1), object(3)
memory usage: 93.9+ KB
```

In [5]:
```
import matplotlib.pyplot as plt

# Assuming 'genre' column exists in the 'books' DataFrame
genre_counts = books['genre'].value_counts()

# Create the pie chart
plt.figure(figsize=(8, 8))  # Adjust figure size as needed
plt.pie(genre_counts, labels=genre_counts.index, autopct='%1.1f%%', startangle=90)
plt.title('Distribution of Book Genres')
plt.axis('equal')  # Equal aspect ratio ensures that the pie chart is circular
plt.show()
```



In [6]:
```python
from IPython.display import HTML

# Assuming 'summary' column exists and is not empty
if 'summary' in books.columns and not books['summary'].isnull().all():
    summary_text = books['summary'].iloc[1]
    if pd.notna(summary_text):  # Check if the summary is not NaN
        colored_summary = f"<div style='text-align:justify; color:blue; font-size:14pt;'>{summary_text}</div>"
        display(HTML(colored_summary))
    else:
        print("The summary at index 1 is empty or NaN.")
else:
    print("The 'summary' column does not exist or is entirely empty/NaN.")
```

As the book opens, Jason awakens on a school bus, unable to remember who or where he is, or anything about his past. He is sitting next to Piper McLean and Leo Valdez, who call him by name and say they are his girlfriend and best friend respectively. All three are part of a class field trip to the Grand Canyon, and after they arrive, a classmate Dylan turns into a Venti (Storm Spirit) and attacks the trio and their trip leader, Coach Gleeson Hedge. In the ensuing fight, Jason surprises everyone, including himself, when one of his coins turns into a sword which he uses to battle the storm spirits. Coach Hedge, who reveals himself to be a satyr during the fight, is taken captive by a fleeing spirit. After the battle, a flying chariot arrives to rescue the trio, but one of the people in it, Annabeth, is upset when she discovers that her missing boyfriend, Percy Jackson, is not there as she expected. Annabeth, seeking Percy, was told in a vision from the goddess Hera to look there for the "guy with one shoe", but this turns out to be Jason, who had a shoe destroyed during the fight. Jason, Piper,and Leo are told that they are demigods and are taken back to Camp Half-Blood where they meet other greek demigod children like themselves. There, Leo is revealed as a son of Hephaestus, Piper as a daughter of Aphrodite and Jason as a son of Zeus, though Hera tells him he is her champion. Jason later discovers that he is the full brother of Zeus's demigod daughter Thalia Grace, who is a Hunter of Artemis. Shortly after they arrive, the three are given a quest to rescue Hera, who has been captured, and they set off. They soon discover that their enemies are working under orders from Gaea to overthrow the gods. During their quest, they encounter Thalia and the Hunters, who have been looking for Percy. Thalia and Jason reunite for the first since Jason was captured at the age of two. On the way to Aeolus's castle, Jason, Leo and Piper become separated from Thalia, who promises to meet them at the Wolf House, the last place Thalia had seen Jason before this meeting. After being nearly apprehended by Aeolus, who is under Gaea's orders, the trio manage to escape thanks to Mellie, Aeolus`s former assistant, and end up in San Francisco, thanks to the result of a dream Piper had with Aphrodite. After landing in San Francisco, the trio rush to Mt.Diablo to fight the giant Enceladus, who has kidnapped Piper's father. They manage to kill the giant and save Piper's father, after which they rush to the Wolf House to free Hera. Although the heroes and the Hunters save Hera, the king of the giants, Porphyrion, rises fully and disappears into a hole in the Earth. Jason's memory then starts returning, and he remembers that he is a hero from a Roman counterpart to Camp Half-Blood somewhere near San Francisco, and is the son of Jupiter, Zeus's Roman aspect. He realizes that Hera, also known as Juno, has switched him with Percy Jackson, who will be at the Roman camp with no memory of his life, in the hopes that the two camps would ultimately work together to fight the giants and defeat the goddess Gaea.

In [7]:
```
from IPython.display import HTML

# Assuming 'summary' column exists and is not empty
if 'summary' in books.columns and not books['summary'].isnull().all():
    summary_text = books['summary'].iloc[200]
    if pd.notna(summary_text):  # Check if the summary is not NaN
        colored_summary = f"<div style='text-align:justify; color:Red; font-size:14pt;'>{summary_text}</div>"
        display(HTML(colored_summary))
    else:
        print("The summary at index 1 is empty or NaN.")
else:
    print("The 'summary' column does not exist or is entirely empty/NaN.")
```

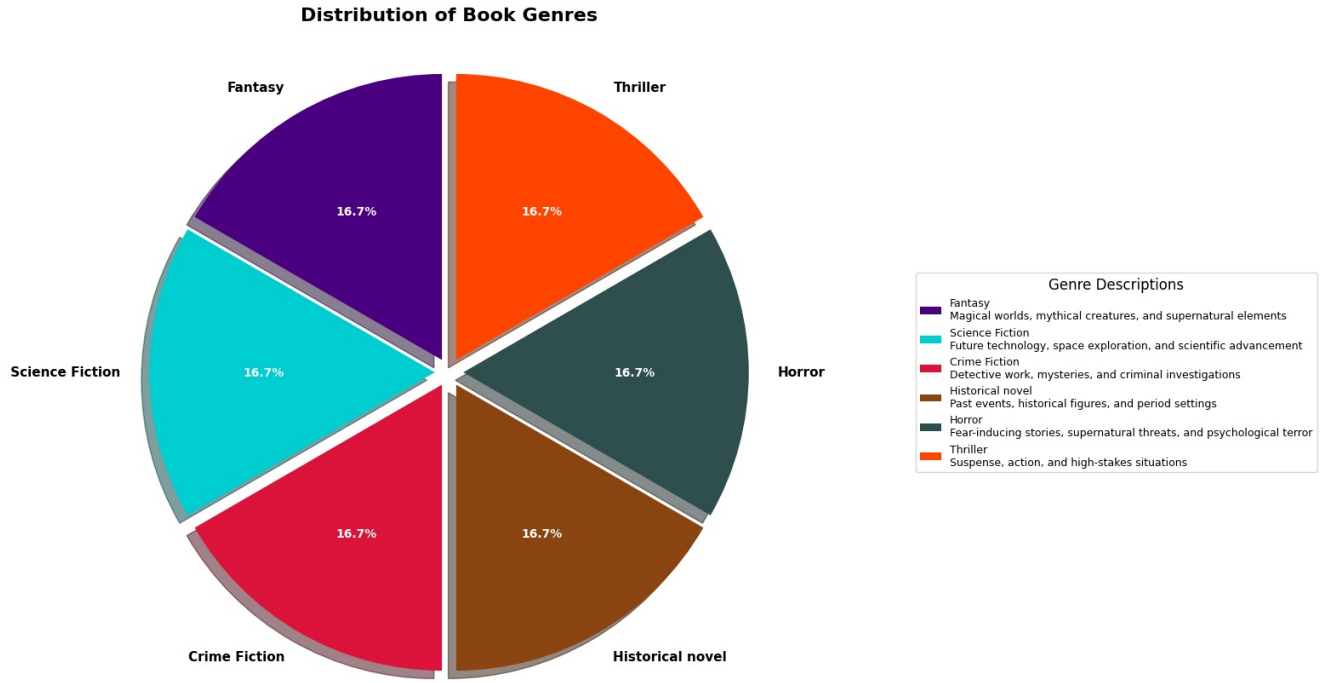
The plot concerns a young woman living in Ottawa named Jacky Rowan who, after a late-night encounter with a motorcycle-riding version of the Wild Hunt, picks up a red cap which enables her to see into the Faerie realms. She is soon drawn into a supernatural struggle between the weakened forces of the Seelie Court and their ominous enemies, the Host or Unseelie Court. She is regaled as the Jack of Kinrowan, a trickster figure who represents the Seelie Court's hope for victory against the forces of evil. With the help of her friend Kate Hazel and an array of faerie friends and allies she makes along the way (and a considerable amount of good luck), Jacky manages to rescue the kidnapped daughter of the Laird of Kinrowan and defeat the Unseelie Court, thus bringing peace and safety to the land.

In [8]:
```
%%time

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from IPython.display import HTML, display
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn import metrics
import nltk
from collections import Counter
import re

class BookAnalyzer:
    def __init__(self, csv_path):
        """Initialize the BookAnalyzer with the dataset path"""
        self.books = pd.read_csv(csv_path)
        if 'Unnamed: 0' in self.books.columns:
            self.books = self.books.drop(['Unnamed: 0'], axis=1)

        # Updated genre colors with richer, more distinctive hex codes
        self.genre_colors = {
            'Fantasy': '#4B0082',           # Deep Indigo - mystical and magical
            'Science Fiction': '#00CED1',    # Dark Turquoise - futuristic and tech-inspired
            'Crime Fiction': '#DC143C',      # Crimson - intense and dramatic
            'Historical novel': '#8B4513',   # Saddle Brown - aged and classic
            'Horror': '#2F4F4F',            # Dark Slate Gray - dark and mysterious
            'Thriller': '#FF4500',          # Orange Red - high energy and suspense
        }

        # Add genre descriptions
        self.genre_descriptions = {
            'Fantasy': 'Magical worlds, mythical creatures, and supernatural elements',
            'Science Fiction': 'Future technology, space exploration, and scientific advancement',
            'Crime Fiction': 'Detective work, mysteries, and criminal investigations',
            'Historical novel': 'Past events, historical figures, and period settings',
            'Horror': 'Fear-inducing stories, supernatural threats, and psychological terror',
            'Thriller': 'Suspense, action, and high-stakes situations'
        }

    def plot_genre_distribution(self):
        """Create an enhanced pie chart of genre distribution with improved styling"""
        genre_counts = self.books['genre'].value_counts()

        plt.figure(figsize=(14, 10))
        colors = [self.genre_colors.get(genre, '#2C3E50') for genre in genre_counts.index]

        wedges, texts, autotexts = plt.pie(genre_counts,
                                           labels=genre_counts.index,
                                           autopct='%1.1f%%',
                                           startangle=90,
                                           colors=colors,
                                           shadow=True,
                                           explode=[0.05] * len(genre_counts))  # Added explode effect

        # Enhanced label and percentage appearance
        plt.setp(autotexts, size=10, weight="bold", color='white')
        plt.setp(texts, size=11, weight="bold")

        plt.title('Distribution of Book Genres', size=16, pad=20, weight='bold')
        plt.axis('equal')

        # Enhanced legend with descriptions
        legend_labels = [f"{genre}\n{self.genre_descriptions[genre]}"
                         for genre in genre_counts.index]
```

```
        plt.legend(wedges, legend_labels,
                  title="Genre Descriptions",
                  loc="center left",
                  bbox_to_anchor=(1, 0, 0.5, 1),
                  fontsize=9,
                  title_fontsize=12)

        plt.show()

    def display_colored_summaries(self, max_books=5):
        """Display book summaries with genre-specific styling and themes"""
        styles = """
        <style>
            .book-container {
                margin: 25px 0;
                padding: 25px;
                border-radius: 12px;
                background-color: #f8f9fa;
                box-shadow: 0 6px 12px rgba(0,0,0,0.1);
                transition: all 0.3s ease;
            }

            /* Fantasy Theme */
            .fantasy-container {
                background: linear-gradient(to right, #f8f9fa, #f0f2ff);
                border-left: 5px solid #4B0082;
                border-right: 2px solid #8A2BE2;
            }
            .fantasy-container:hover {
                transform: translateY(-5px) rotate(0.5deg);
                box-shadow: 0 8px 20px rgba(75,0,130,0.2);
            }
            .fantasy-container .genre-label {
                text-shadow: 1px 1px 2px rgba(75,0,130,0.3);
            }

            /* Science Fiction Theme */
            .scifi-container {
                background: linear-gradient(to right, #f8f9fa, #f0ffff);
                border-left: 5px solid #00CED1;
                border-bottom: 2px solid #40E0D0;
            }
            .scifi-container:hover {
                transform: translateY(-5px) translateX(3px);
                box-shadow: 0 8px 20px rgba(0,206,209,0.2);
            }
            .scifi-container .metadata {
                font-family: 'Courier New', monospace;
            }

            /* Crime Fiction Theme */
            .crime-container {
                background: linear-gradient(to right, #f8f9fa, #fff5f5);
                border-left: 5px solid #DC143C;
                border-top: 2px solid #FF6B6B;
            }
            .crime-container:hover {
                transform: translateY(-5px) skew(-1deg);
                box-shadow: 0 8px 20px rgba(220,20,60,0.2);
            }
            .crime-container .summary-text {
                text-indent: 30px;
            }

            /* Historical Novel Theme */
            .historical-container {
                background: linear-gradient(to right, #f8f9fa, #fff9f0);
                border-left: 5px solid #8B4513;
                border: 2px solid #DEB887;
                border-left: 5px solid #8B4513;
            }
            .historical-container:hover {
                transform: translateY(-5px);
                box-shadow: 0 8px 20px rgba(139,69,19,0.2);
            }
            .historical-container .genre-description {
                font-family: 'Georgia', serif;
            }

            /* Horror Theme */
            .horror-container {
                background: linear-gradient(to right, #f8f9fa, #f2f2f2);
                border-left: 5px solid #2F4F4F;
                border-right: 2px solid #4A4A4A;
            }
            .horror-container:hover {
                transform: translateY(-5px) scale(1.01);
                box-shadow: 0 8px 20px rgba(47,79,79,0.3);
```

```
            }
            .horror-container .genre-label {
                letter-spacing: 3px;
            }

            /* Thriller Theme */
            .thriller-container {
                background: linear-gradient(to right, #f8f9fa, #fff5f0);
                border-left: 5px solid #FF4500;
                border-bottom: 2px solid #FF6347;
            }
            .thriller-container:hover {
                transform: translateY(-5px) translateX(-3px);
                box-shadow: 0 8px 20px rgba(255,69,0,0.2);
            }
            .thriller-container .summary-text {
                line-height: 2;
            }

            /* Common Styles */
            .genre-label {
                font-weight: bold;
                font-size: 20px;
                margin-bottom: 15px;
                text-transform: uppercase;
                letter-spacing: 1.5px;
                color: var(--genre-color);
            }
            .genre-description {
                font-style: italic;
                color: #666;
                margin-bottom: 15px;
                font-size: 14px;
            }
            .summary-text {
                text-align: justify;
                font-size: 16px;
                line-height: 1.8;
                font-family: 'Arial', sans-serif;
                color: #333;
            }
            .metadata {
                margin-top: 15px;
                font-size: 13px;
                color: #666;
                border-top: 1px solid #eee;
                padding-top: 10px;
            }
        </style>
        """
        display(HTML(styles))

        genre_classes = {
            'Fantasy': 'fantasy-container',
            'Science Fiction': 'scifi-container',
            'Crime Fiction': 'crime-container',
            'Historical novel': 'historical-container',
            'Horror': 'horror-container',
            'Thriller': 'thriller-container'
        }

        displayed_count = 0
        for idx, row in self.books.iterrows():
            if displayed_count >= max_books:
                break

            summary = row.get('summary')
            genre = row.get('genre')

            if pd.notna(summary) and pd.notna(genre):
                color = self.genre_colors.get(genre, '#2C3E50')
                description = self.genre_descriptions.get(genre, '')
                container_class = genre_classes.get(genre, '')

                html_content = f"""
                <div class='book-container {container_class}' style='--genre-color: {color};'>
                    <div class='genre-label'>
                        {genre}
                    </div>
                    <div class='genre-description'>
                        {description}
                    </div>
                    <div class='summary-text'>
                        {summary}
                    </div>
                    <div class='metadata'>
                        <span style='color: {color};'></span> Book Index: {idx + 1}
                        {f" | ✍ Author: {row.get('author', 'Unknown')}" if 'author' in self.books.columns else
                        {f" | Year: {row.get('year', 'Unknown')}" if 'year' in self.books.columns else ""}
```

```
                    </div>
                </div>
                """
                display(HTML(html_content))
                displayed_count += 1

        if displayed_count == 0:
            print("No valid summaries found to display.")

def main():
    # Initialize analyzer with your dataset
    analyzer = BookAnalyzer('/kaggle/input/bookdata/BooksDataSet.csv')

    # Create genre distribution visualization
    analyzer.plot_genre_distribution()

    # Display colored summaries
    analyzer.display_colored_summaries(max_books=5)

if __name__ == "__main__":
    main()
```



FANTASY

Magical worlds, mythical creatures, and supernatural elements

Drowned Wednesday is the first Trustee among the Morrow Days who is on Arthur's side and wishes the Will to be fulfilled. She appears as a leviathan/whale and suffers from Gluttony. The book begins when Leaf is visiting Arthur and they are discussing the invitation that Drowned Wednesday sent him. Arthur had been admitted to hospital because of the damage done to his leg when he attempted to enter Tuesday's Treasure Tower. Suddenly, the hospital room becomes flooded with water as the two are transported to the Border Sea of the House. Leaf is snatched away by a large ship with green sails, known as the Flying Mantis, while Arthur remains in his bed. When the Medallion given him by the immortal called the Mariner apparently fails to summon help, Arthur is without hope. Eventually, a buoy marking the pirate Elishar Feverfew's treasure floats toward him. As soon as Arthur opens it, his hand is marked with a bloody red colour. Arthur now has the Red Hand, by which Feverfew marks whoever has found his treasure, so that he can identify them later. Not long after, a scavenging ship called the Moth rescues Arthur. On board, Arthur (going by the name of Arth) is introduced to Sunscorch, the First Mate, and to Captain Catapillow. Their journey brings them through the Line of Storms and into the Border Sea, where they are later pursued by Feverfew's ghostly ship, the Shiver. The damage inflicted on the Moth is serious; therefore Sunscorch commands an Upper House Sorcerer, Dr. Scamandros, to open a transfer portal to elsewhere in the Secondary Realms. Scamandros claims that Arthur is carrying something that interfered with his magic, and tells Sunscorch to throw him overboard. As a last resort, Arthur shows them the

Mariner's Medallion, which stops Scamandros saying that they must get rid of Arthur. After going through the transfer portal (with Arthur's help), the ship is grounded on a beach. When Arthur wished to learn what happened to Leaf, Dr. Scamandros applies his sorcery to make it possible. She is revealed to be aboard the ship Flying Mantis. Arthur joins Catapillow for supper, later to reveal his identity. At first, Propaganda issued by Dame Primus (Arthur's Steward) makes them skeptical of this, but they eventually become convinced. A few days later, Wednesday's Dawn takes Arthur to meet Wednesday for her 'luncheon of seventeen removes'. As they approach, Wednesday shrinks into her human form to meet Arthur. During their lunch, Wednesday tells Arthur that after Part 3 of the Will has been released, she will surrender the Third Key to Arthur. Arthur is then taken by Wednesday's Dawn to a place called the triangle in search of his friend Leaf. He learns that Leaf has been forced to work on the Mantis, but is otherwise intact. Arthur later makes a deal with the Raised Rats, a group of anthropomorphic rats brought to the House by the Piper, to take him to Feverfew's hideout, which they believe is inside a miniature world located within Drowned Wednesday's stomach. On the Raised Rats' ship, Arthur opens a gift from Dr. Scamandros, which proves to be a golden transfer watch. With this, he communicates with and rescues Dr. Scamandros. He then uses a scrying mirror Dr. Scamandros gave him to watch Leaf. A rat watches him during the scry, and later saves him from a battle with Feverfew. Later, the Rats bring Arthur onto their submarine, where he meets with Suzy Turquiose Blue. In contrast to Suzy's former cockney attitude, she has assumed a more "ladylike and proper" demeanor on the orders of Dame Primus. Only when they are no longer on the Border Sea, but under it, does she resume her customary ways of speech and dress. They are, with navigational difficulty, able to enter the stomach of Lady Wednesday and the worldlet therein. There, Arthur and Suzy, disguised as rats, find escaped slaves, professed followers of the Carp. These exiles take them to the Carp, who is the third part of the Will. They are halted in their attempt to escape by Feverfew, who proposes that each of them will try to kill the other by means of one strike only. Arthur fails his first try, then dodges Feverfew and severs his head. Leaf, who is Feverfew's prisoner, then kicks it into a mud puddle containing Nothing, which consumes it. Upon his death, the worldlet begins to collapse. Via the Moth, Arthur and all his friends (with the exception of the reluctant Catapillow) are able to escape. Lady Wednesday recovers from her gluttony, then dies as a result of being poisoned by the worldlet, which had opened a void to Nothing. Arthur, now Duke of the Border Sea, appoints Sunscorch as his Noon and Scamandros as his Dusk. fr:Mercredi sous les flots th:พุธเพชฌฆาต

Book Index: 1

FANTASY

Magical worlds, mythical creatures, and supernatural elements

As the book opens, Jason awakens on a school bus, unable to remember who or where he is, or anything about his past. He is sitting next to Piper McLean and Leo Valdez, who call him by name and say they are his girlfriend and best friend respectively. All three are part of a class field trip to the Grand Canyon, and after they arrive, a classmate Dylan turns into a Venti (Storm Spirit) and attacks the trio and their trip leader, Coach Gleeson Hedge. In the ensuing fight, Jason surprises everyone, including himself, when one of his coins turns into a sword which he uses to battle the storm spirits. Coach Hedge, who reveals himself to be a satyr during the fight, is taken captive by a fleeing spirit. After the battle, a flying chariot arrives to rescue the trio, but one of the people in it, Annabeth, is upset when she discovers that her missing boyfriend, Percy Jackson, is not there as she expected. Annabeth, seeking Percy, was told in a vision from the goddess Hera to look there for the "guy with one shoe", but this turns out to be Jason, who had a shoe destroyed during the fight. Jason, Piper,and Leo are told that they are demigods and are taken back to Camp Half-Blood where they meet other greek demigod children like themselves. There, Leo is revealed as a son of Hephaestus, Piper as a daughter of Aphrodite and Jason as a son of Zeus, though Hera tells him he is her champion. Jason later discovers that he is the full brother of Zeus's demigod daughter Thalia Grace, who is a Hunter of Artemis. Shortly after they arrive, the three are given a quest to rescue Hera, who has been captured, and they set off. They soon discover that their enemies are working under orders from Gaea to overthrow the gods. During their quest, they encounter Thalia and the Hunters, who have been looking for Percy. Thalia and Jason reunite for the first since Jason was captured at the age of two. On the way to Aeolus's castle, Jason, Leo and Piper become separated from Thalia, who promises to meet them at the Wolf House, the last place Thalia had seen Jason before this meeting. After being nearly apprehended by Aeolus, who is under Gaea's orders, the trio manage to escape thanks to Mellie, Aeolus`s former assistant, and end up in San Francisco, thanks to the result of a dream Piper had with Aphrodite. After landing in San Francisco, the trio rush to Mt.Diablo to fight the giant Enceladus, who has kidnapped Piper's father. They manage to kill the giant and save Piper's father, after which they rush to the Wolf House to free Hera. Although the heroes and the Hunters save Hera, the king of the giants, Porphyrion, rises fully and disappears into a hole in the Earth. Jason's memory then starts returning, and he remembers that he is a hero from a Roman counterpart to Camp Half-Blood somewhere near San Francisco, and is the son of Jupiter, Zeus's Roman aspect. He realizes that Hera, also known as Juno, has switched him with Percy Jackson, who will be at the Roman camp with no memory of his life, in the hopes that the two camps would ultimately work together to fight the giants and defeat the goddess Gaea.

FANTASY

Magical worlds, mythical creatures, and supernatural elements

Cugel is easily persuaded by the merchant Fianosther to attempt the burglary of the manse of Iucounu the Laughing Magician. Trapped and caught, he agrees that in exchange for his freedom he will undertake the recovery of a small hemisphere of violet glass, an Eye of the Overworld, to match one already in the wizard's possession. A small sentient alien entity of barbs and hooks, named Firx, is attached to his liver to encourage his "unremitting loyalty, zeal and singleness of purpose," and Iucounu uses a spell to transport Cugel via flying demon to the remote Land of Cutz. There, Cugel finds two villages, one occupied by wearers of the violet lenses, the other by peasants who work on behalf of the lens-wearers, in hopes of being promoted to their ranks. The lenses cause their wearers to see, not their squalid surroundings, but the Overworld, a vastly superior version of reality where a hut is a palace, gruel is a magnificent feast, etc. — "seeing the world through rose-colored glasses" on a grand scale. Cugel gains an Eye by trickery, and escapes from Cutz. He then undertakes an arduous trek back to Iucounu, cursing the magician the entire way; this forms the principal part of the book. After many pitfalls, setbacks, and harrowing escapes, including the eviction of Firx from his system, Cugel returns to Iucounu's manse, where he finds the wizard's volition has been captured by a twin to Firx. Cugel manages to extirpate the alien, subdue the magician, and enjoy the easy life in the manse, until he tries to banish Iucounu and Fianosther (who himself has come to pilfer from Cugel) with the same spell that the magician had used on him. But Cugel's tongue slips in uttering the incantation, and the flying demon seizes him instead, delivering him to the same spot as before. Author Michael Shea wrote an authorized sequel, A Quest for Simbilis (DAW Books, NY, 1974). Vance's own Cugel sequel was published as Cugel's Saga in 1983.

FANTASY

Magical worlds, mythical creatures, and supernatural elements

The book opens with Herald-Mage Vanyel returning to his country Valdemar from an extensive campaign along the border with Karse, the neighboring enemy country. He checks in with Valdemar's King Randale and his lifebonded mate, Shavri, and their daughter Jisa. Only a select few know that Jisa is Vanyel's daughter. Because King Randale is sterile, he had asked Vanyel to father an heir. Now, Shavri confides in Vanyel her fear that Randale is mortally ill. Vanyel and his mentor Savil return to Vanyel's family home at Forst Reach, where they find little rest or peace. His parents both try to change his mind about being shay'a'chern, or homosexual. Vanyel becomes somewhat confused about his own sexuality. He wonders if he is truly in love with Shavri. Yfandes, Vanyel's Companion, doesn't buy it and finds the situation amusing. Vanyel also meets his illegitimate nephew, Medren. Medren is small for his age as Vanyel was, and like him it appears that he is often bullied by the armsmaster Jervis. The boy has a powerful Bardic gift, and Vanyel sponsors him for the Bardic Collegium. Vanyel confronts Jervis and learns that the armsmaster is not being intentionally rough. He also apologizes for beating Vanyel long ago, and explains his difficult position with Vanyel's father. He also mentions that he knew Vanyel was shay'a'chern from the beginning, but also knew from his army service that being gay does not keep men from being courageous warriors. Vanyel accepts this and they form an uneasy friendship. The main plot focuses on how Vanyel assists young Prince Tashir Remoerdis. Vanyel and Yandes receive a psychic summons into the neighboring country of Lineas. Upon arrival, they find Tashir, who has just become a Herald, with his Companion, Leshya (nicknamed Ghost by Yfandes). Another Herald, Lores, is beating the child and attacks Ghost when he tries to interfere. He believes that Tashir is actually an evil sorcerer who has murdered his own family. Vanyel immediately stops Lores and commands him to return to Valdemar. Vanyel takes Tashir back to his own home, Forst Reach, and begins to try and discover what truly happened to Tashir's family. He then returns to Lineas, disguised as a minstrel to gather information. In Lineas magic is taboo; no one is supposed to perform it for any reason. It shares a border with Baires, a country ruled by the Mavelan, a family of mages. A treaty was signed to end the warring between the two countries, sealed by the marriage of Tashir's parents. In the treaty there exists a clause that if the royal family of Lineas dies out, the Mavelan can take over the Lineas throne, and vice versa. Vanyel meets an old servant of the Remoerdis family who tells him of Tashir's sad childhood. Tashir was physically and emotionally abused by his father and sexually abused by his mother. His mother was mentally unstable and made seductive overtures to Tashir even as a young child. Tashir looks very much like his mother's uncle Vedric, and rumor has it that he is the product of an incestuous affair between them. Returning to Forst Reach, Vanyel recruits Savil and Jervis to help. Together with Tashir, they return to Lineas, breaking through the palace's magic shields to search the place. They find a secret room which contains a "heart-stone", an ordinary rock connected with a magical "node" which is keeping a deep, dangerous fault sealed. As the stone was unstable and any magic done in the area could disturb it, a guardian family without magic was appointed. If the stone were to be removed, a giant earthquake would occur destroying Lineas, parts of the outlands, and the border of Valdemar. That was why magic was anathema in Lineas, and also why everyone in the palace was blood related member of the Remoerdis family, sworn to protect the node. In the palace Vanyel and Savil discover a "trap-spell", which targets one person and also kills the target's entire family. Tashir's family were the victims of the trap-spell, which was placed by Tashir's uncle Vedric, whose intention is to control both Lineas and Baires. (He is not, however, Tashir's father.) Vedric tries the same trick on Vanyel, who has already caught on and sent Savil away, along with Tashir, while keeping Jervis with him. In defending himself, Vanyel ends up killing most of the Baires family as well as Vedric. Now king of both Baires and Lineas, Tashir considers himself too young and inexperienced. He allows King Randale to annex the two smaller countries into Valdemar. Vanyel finally sorts out that he actually is gay, and that Shavri is just a good friend.

FANTASY

Magical worlds, mythical creatures, and supernatural elements

Taran and Gurgi have returned to Caer Dallben following the events of the third chronicle, The Castle of Llyr. It is now full springtime, at least three years after The Book of Three. Taran knows that he loves Princess Eilonwy, who now resides at Dinas Rhydnant for royal education. In full springtime some weeks after return from escorting her there, he is restless and determines to know his parentage, noble or common, partly in hopes that noble birth will support a marriage proposal. Dallben the enchanter tells him nothing but gives his approval for Taran and Gurgi to travel on their own. They travel first to the Marshes of Morva to ask the witches Orddu, Orwen and Orgoch. Taran has nothing of great value to give in exchange, so Orddu merely tells him of an alternative, the Mirror of Llunet in the far east Llawgadarn Mountains will show him who he is. Taran goes next to Cantrev Cadiffor to be outfit by King Smoit for a longer journey. After a border patrol of Smoit's vassal Lord Goryon steals his horse Melynlas and Gurgi's pony, they spend the night with the farm couple Aeddan and Alarca who have lost their son and livestock. Taran would be welcome to remain and he leaves with new respect for common farmers. They recover their steeds because Melynlas will have no other rider and Goryon is relieved to escape the honorary burden of mastering him.At the neighboring stronghold of Lord Gast they meet old friend Fflewddur Fflam, who seems to return to wandering as a bard every spring. Together they go on to Caer Cadarn where Smoit does welcome them. Goryon and Gast have been feuding for years (every spring?), especially over the prize cow Cornillo. When their dispute breaks out again next day, Taran questions King Smoit's habitual resort to imprisonment and persuades Smoit to try Taran's judgment. Namely, the rival cantrev lords shall resow the fields of Aeddan, which have been ruined by battles. The prize cow shall be further compensation, although the lords shall have her calves. The remainder of their commingled herds shall be divided in half by Goryon, and Gast will choose which half to take for himself. The childless widower Smoit later offers to adopt Taran as his son, who will succeed as King of Cadiffor. Taran declines but says he will gladly accept if he discovers noble birth. Continuing eastward, they cross Ystrad. Taran's pet crow Kaw and Fflewddur's mount Llyan, a giant lion, find treasures which they bring to their masters. Kaw a polished bone the size of a toothpick, which has been stashed high in a tree. Llyan a green and yellow frog that has nearly dried to death: their old friend Doli the dwarf, whom they revive. Doli has been transformed during investigation of a deadly threat to the Fair Folk. He knows that he is not the first to disappear in animal form, for a human wizard Morda has attained power to enchant them, and to raid their underground realms. Taran and Gurgi investigate Morda's abode, followed by Fflewddur, but all are captured by his snares. After explaining himself (history, boasts, plans), Morda turns Fflewddur and Gurgi into a hare and a mouse, but fails to transform Taran. Something protects him, and he guesses from the stump of Morda's little finger that it is the polished bone. Although elderly, Morda is stronger than Taran, but his strength finally snaps the bone in desperate fury to regain it. (Morda has worn a silver crescent moon with pendant jewel on a necklace. Eilonwy has one without the jewel, and Morda prompted by Taran's recognition of the symbol of the House of Llyr has revealed the primary source of his magic. After Eilonwy was kidnapped as a child, the long search by her mother Angharad ended here, where she weakly sought shelter. Morda inherited both the amulet she wore and the empty book among her possessions. He mastered the amulet and developed its power, gave the book to a pest Glew. (The Castle of Llyr.) Morda's death restores Doli and others to their natural forms. Before parting, Taran gives the jewel from Angharad's amulet to him (returning a gift by the Fair Folk to the House of Llyr) and Doli identifies the ceremonial horn Taran wears in token of Eilonwy's pledge. It will summon the Fair Folk to his assistance, and one summons remains. Taran, Gurgi, and Fflewddur camp next with the ruffian Dorath and his band. Their hosts suspect a quest for treasure and offer guidance to Llunet, in exchange for a share. The guests try to slip away early next morning but Dorath prevents that and extracts a wager on hand-to-hand combat with Taran. He cheats and takes Taran's sword (the stake), then departs. An old shepherd Craddoc, with decrepit holdings, welcomes the companions next. From Taran's account of the mission, he welcomes Taran as his son. Fflewddur departs but Taran and Gurgi remain and labor beside him. Taran and Craddoc develop some bond, but Taran also resents the end of his dream of noble birth. During the next winter, however, Craddoc suffers a bad fall down a mountain gorge, and Taran is unable to rescue him. Near death he reveals that he knew enough of Dallben's history to pose as father and gain Taran as a son. The gorge and the

weather threaten Taran too, and he finally summons the Fair Folk who are able to save only himself and Gurgi. Afterward Taran and Gurgi continue eastward, across Little Avren to the Free Commots. They stay first with lucky Llonio and his family on the banks of the river, where Taran learns that life is a net to gather what comes. Next Taran assists and learns the trades of three great craftmasters: Hevydd the smith, Dwyvach the weaver, and Annlaw the potter. They teach him that life is a forge, a loom, and a potter's wheel. He learns enough that he would be welcome to remain as assistance, but pottery alone seems to call him and he realizes to his dismay that he will never master it. He does have a new sword, new cloak, and new bowl. Driving the wares of Annlaw to Commot Ivar, he leads the poor farming village in resistance to a raid by Dorath, killing half the band at no loss of life. Annlaw tells Taran the way to the Mirror of Llunet; he knows it, but has never visited, for he knows who he is. After a short journey, Taran and Gurgi find the Mirror, a pool of water at the mouth of a cave beyond the Lake of Llunet. Taran gazes into it, and he is astonished, but Dorath interrupts, now alone. Evidently there is no treasure, so Taran and the ruffian are soon at swordpoint. His old sword shatters on his new one and Dorath flees. Taran does not pursue but returns to Annlaw Clay-Shaper. He relates that the Mirror showed his own reflection and nothing more. Cheated by Orddu? No, for he saw what he had become and all he had learned on the way.

Book Index: 5

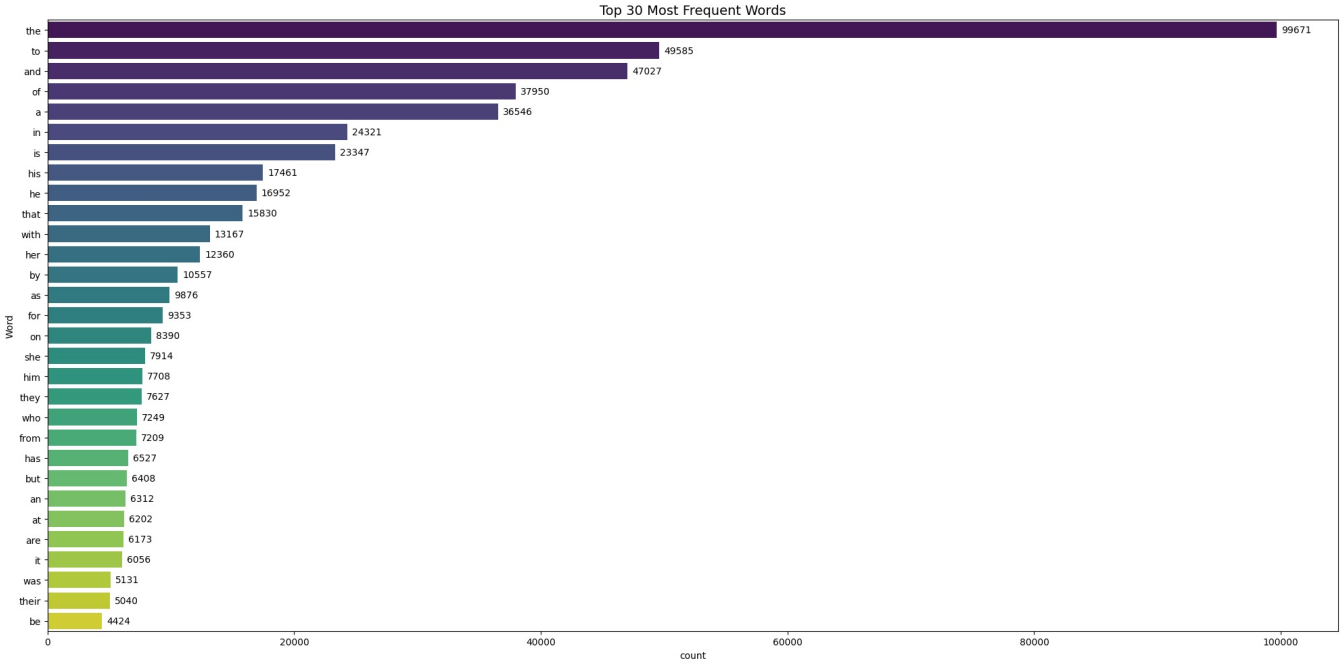
```
CPU times: user 593 ms, sys: 220 ms, total: 813 ms
Wall time: 552 ms
```

In [9]:
```python
import re

def clean_text(text):
    # Remove special symbols and convert text to lowercase in one step
    text = re.sub(r"[^a-zA-Z\s]", "", text).lower()
    # Remove extra whitespaces
    text = ' '.join(text.split())
    return text

books['summary'] = books['summary'].apply(clean_text)
books['summary'].iloc[1]
```

Out[9]:
```
'as the book opens jason awakens on a school bus unable to remember who or where he is or anything about his pa
st he is sitting next to piper mclean and leo valdez who call him by name and say they are his girlfriend and b
est friend respectively all three are part of a class field trip to the grand canyon and after they arrive a cl
assmate dylan turns into a venti storm spirit and attacks the trio and their trip leader coach gleeson hedge in
the ensuing fight jason surprises everyone including himself when one of his coins turns into a sword which he
uses to battle the storm spirits coach hedge who reveals himself to be a satyr during the fight is taken captiv
e by a fleeing spirit after the battle a flying chariot arrives to rescue the trio but one of the people in it
annabeth is upset when she discovers that her missing boyfriend percy jackson is not there as she expected anna
beth seeking percy was told in a vision from the goddess hera to look there for the guy with one shoe but this
turns out to be jason who had a shoe destroyed during the fight jason piperand leo are told that they are demig
ods and are taken back to camp halfblood where they meet other greek demigod children like themselves there leo
is revealed as a son of hephaestus piper as a daughter of aphrodite and jason as a son of zeus though hera tell
s him he is her champion jason later discovers that he is the full brother of zeuss demigod daughter thalia gra
ce who is a hunter of artemis shortly after they arrive the three are given a quest to rescue hera who has been
captured and they set off they soon discover that their enemies are working under orders from gaea to overthrow
the gods during their quest they encounter thalia and the hunters who have been looking for percy thalia and ja
son reunite for the first since jason was captured at the age of two on the way to aeoluss castle jason leo and
piper become separated from thalia who promises to meet them at the wolf house the last place thalia had seen j
ason before this meeting after being nearly apprehended by aeolus who is under gaeas orders the trio manage to
escape thanks to mellie aeoluss former assistant and end up in san francisco thanks to the result of a dream pi
per had with aphrodite after landing in san francisco the trio rush to mtdiablo to fight the giant enceladus wh
o has kidnapped pipers father they manage to kill the giant and save pipers father after which they rush to the
wolf house to free hera although the heroes and the hunters save hera the king of the giants porphyrion rises f
ully and disappears into a hole in the earth jasons memory then starts returning and he remembers that he is a
hero from a roman counterpart to camp halfblood somewhere near san francisco and is the son of jupiter zeuss ro
man aspect he realizes that hera also known as juno has switched him with percy jackson who will be at the roma
n camp with no memory of his life in the hopes that the two camps would ultimately work together to fight the g
iants and defeat the goddess gaea'
```

In [10]:
```python
import warnings
warnings.filterwarnings('ignore')


def showmostfrequentwords(text, no_of_words):
    allwords = ' '.join([char for char in text])
    allwords = allwords.split()
    fdist = nltk.FreqDist(allwords)
    wordsdf = pd.DataFrame({'word': list(fdist.keys()), 'count': list(fdist.values())})
    df = wordsdf.nlargest(columns="count", n=no_of_words)
    plt.figure(figsize=(20, 10))  # Increased figure size for better readability
    ax = sn.barplot(data=df, x='count', y='word', palette="viridis")  # Using a different color palette
    ax.set(ylabel='Word')
    ax.set_title(f"Top {no_of_words} Most Frequent Words", fontsize=14)  # Added title
```

```
    # Annotate bars with their count values
    for p in ax.patches:
        ax.annotate(format(p.get_width(), '.0f'),
                    (p.get_width(), p.get_y() + p.get_height() / 2.),
                    ha='left', va='center',
                    xytext=(5, 0), textcoords='offset points', fontsize=10) # Adjusted for visibility

    plt.tight_layout() # Adjust layout to prevent labels from overlapping
    plt.show()
    return wordsdf

wordsdf = showmostfrequentwords(books['summary'], 30)
```



In [11]:

```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords

# Set of English stopwords
stop_words = set(stopwords.words('english'))

# Function to remove stopwords
def remove_stopwords(text):
    return ' '.join([word for word in text.split() if word not in stop_words])

# Apply the function to the 'summary' column
books['summary'] = books['summary'].apply(remove_stopwords)
books['summary'].iloc[3]

if 'summary' in books.columns and not books['summary'].isnull().all():
    summary_text = books['summary'].iloc[3] # Access the summary at index 1
    if pd.notna(summary_text):  # Check if the summary is not NaN
        colored_summary = f"<div style='text-align:justify; color:blue; font-size:14pt;'>{summary_text}</div>"
        display(HTML(colored_summary))
    else:
        print("The summary at index 1 is empty or NaN.")
else:
    print("The 'summary' column does not exist or is entirely empty/NaN.")
```

```
[nltk_data] Downloading package stopwords to /usr/share/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

book opens heraldmage vanyel returning country valdemar extensive campaign along border karse neighboring enemy country checks valdemars king randale lifebonded mate shavri daughter jisa select know jisa vanyels daughter king randale sterile asked vanyel father heir shavri confides vanyel fear randale mortally ill vanyel mentor savil return vanyels family home forst reach find little rest peace parents try change mind shayachern homosexual vanyel becomes somewhat confused sexuality wonders truly love shavri yfandes vanyels companion doesnt buy finds situation amusing vanyel also meets illegitimate nephew medren medren small age vanyel like appears often bullied armsmaster jervis boy powerful bardic gift vanyel sponsors bardic collegium vanyel confronts jervis learns armsmaster intentionally rough also apologizes beating vanyel long ago explains difficult position vanyels father also mentions knew vanyel shayachern beginning also knew army service gay keep

men courageous warriors vanyel accepts form uneasy friendship main plot focuses vanyel assists young prince tashir remoerdis vanyel yandes receive psychic summons neighboring country lineas upon arrival find tashir become herald companion leshya nicknamed ghost yfandes another herald lores beating child attacks ghost tries interfere believes tashir actually evil sorcerer murdered family vanyel immediately stops lores commands return valdemar vanyel takes tashir back home forst reach begins try discover truly happened tashirs family returns lineas disguised minstrel gather information lineas magic taboo one supposed perform reason shares border baires country ruled mavelan family mages treaty signed end warring two countries sealed marriage tashirs parents treaty exists clause royal family lineas dies mavelan take lineas throne vice versa vanyel meets old servant remoerdis family tells tashirs sad childhood tashir physically emotionally abused father sexually abused mother mother mentally unstable made seductive overtures tashir even young child tashir looks much like mothers uncle vedric rumor product incestuous affair returning forst reach vanyel recruits savil jervis help together tashir return lineas breaking palaces magic shields search place find secret room contains heartstone ordinary rock connected magical node keeping deep dangerous fault sealed stone unstable magic done area could disturb guardian family without magic appointed stone removed giant earthquake would occur destroying lineas parts outlands border valdemar magic anathema lineas also everyone palace blood related member remoerdis family sworn protect node palace vanyel savil discover trapspell targets one person also kills targets entire family tashirs family victims trapspell placed tashirs uncle vedric whose intention control lineas baires however tashirs father vedric tries trick vanyel already caught sent savil away along tashir keeping jervis defending vanyel ends killing baires family well vedric king baires lineas tashir considers young inexperienced allows king randale annex two smaller countries valdemar vanyel finally sorts actually gay shavri good friend

In [12]:
```python
import nltk
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
import pandas as pd

# Download required NLTK data
nltk.download('punkt', quiet=True)
nltk.download('stopwords', quiet=True)
nltk.download('omw-1.4', quiet=True)

class TextStemmer:
    def __init__(self, remove_stopwords=False):
        """
        Initialize stemmer with optional stopword removal.

        Args:
            remove_stopwords (bool): Whether to remove stopwords during processing
        """
        self.stemmer = PorterStemmer()
        self.stem_cache = {}
        self.remove_stopwords = remove_stopwords
        if remove_stopwords:
            self.stop_words = set(stopwords.words('english'))
```

```
    def stem_text(self, text):
        """
        Stem text efficiently using caching and optimized string joining.

        Args:
            text (str): Input text to stem

        Returns:
            str: Stemmed text
        """
        if not isinstance(text, str):
            return ""

        # Tokenize and convert to lowercase once
        words = word_tokenize(text.lower())

        # Filter stopwords if enabled
        if self.remove_stopwords:
            words = [word for word in words if word not in self.stop_words]

        # Use list comprehension with cache lookup for stemming
        stemmed_words = [
            self.stem_cache.setdefault(word, self.stemmer.stem(word))
            for word in words
        ]

        # Join words efficiently
        return ' '.join(stemmed_words)

    def process_dataframe(self, df, column):
        """
        Process an entire dataframe column efficiently.

        Args:
            df (pd.DataFrame): Input dataframe
            column (str): Column name to process

        Returns:
            pd.Series: Series with stemmed text
        """
        return df[column].astype(str).apply(self.stem_text)

    def get_example(self, df, column, index):
        """
        Get a specific example from the processed dataframe.

        Args:
            df (pd.DataFrame): Input dataframe
            column (str): Column name
            index (int): Row index

        Returns:
            str: Processed text at specified index
        """
        return df[column].iloc[index]

# Usage example
stemmer = TextStemmer(remove_stopwords=True)  # Initialize with stopword removal

# Process the entire dataframe
books['summary'] = stemmer.process_dataframe(books, 'summary')

# Get specific example
example = stemmer.get_example(books, 'summary', 1)
print(f"Processed example: {example}")
```

```
Processed example: book open jason awaken school bu unabl rememb anyth past sit next piper mclean leo valdez ca
ll name say girlfriend best friend respect three part class field trip grand canyon arriv classmat dylan turn v
enti storm spirit attack trio trip leader coach gleeson hedg ensu fight jason surpris everyon includ one coin t
urn sword use battl storm spirit coach hedg reveal satyr fight taken captiv flee spirit battl fli chariot arriv
rescu trio one peopl annabeth upset discov miss boyfriend perci jackson expect annabeth seek perci told vision
goddess hera look guy one shoe turn jason shoe destroy fight jason piperand leo told demigod taken back camp ha
lfblood meet greek demigod children like leo reveal son hephaestu piper daughter aphrodit jason son zeu though
hera tell champion jason later discov full brother zeuss demigod daughter thalia grace hunter artemi shortli ar
riv three given quest rescu hera captur set soon discov enemi work order gaea overthrow god quest encount thali
a hunter look perci thalia jason reunit first sinc jason captur age two way aeoluss castl jason leo piper becom
separ thalia promis meet wolf hous last place thalia seen jason meet nearli apprehend aeolu gaea order trio man
ag escap thank melli aeoluss former assist end san francisco thank result dream piper aphrodit land san francis
co trio rush mtdiablo fight giant enceladu kidnap piper father manag kill giant save piper father rush wolf hou
s free hera although hero hunter save hera king giant porphyrion rise fulli disappear hole earth jason memori s
tart return rememb hero roman counterpart camp halfblood somewher near san francisco son jupit zeuss roman aspe
ct realiz hera also known juno switch perci jackson roman camp memori life hope two camp would ultim work toget
h fight giant defeat goddess gaea
```

In [13]:

```
books['genre'] = pd.factorize(books['genre'])[0]
books
```

Out[13]:

	book_id	book_name	genre	summary
0	3248537	Drowned Wednesday	0	drown wednesday first truste among morrow day ...
1	27796919	The Lost Hero	0	book open jason awaken school bu unabl rememb ...
2	3910776	The Eyes of the Overworld	0	cugel easili persuad merchant fianosth attempt...
3	5969644	Magic's Promise	0	book open heraldmag vanyel return countri vald...
4	3173445	Taran Wanderer	0	taran gurgi return caer dallben follow event t...
...	...	...	...	...
2995	10372180	White Death	5	novel numa file kurt austin adventur novel mai...
2996	14504372	Venus with Pistol	5	gilbert kemp dealer special antiqu gun london ...
2997	3617412	Blackwater	5	know your deep davey alway live shadow older b...
2998	11320975	The Rainbow and the Rose	5	stori concern life johnni pasco retir commerci...
2999	17227674	Chiefs	5	first chief henri lee novel open grow town del...

3000 rows × 4 columns

In [14]:
```
books['genre'] = pd.factorize(books['genre'])[0]
books
```

Out[14]:

	book_id	book_name	genre	summary
0	3248537	Drowned Wednesday	0	drown wednesday first truste among morrow day ...
1	27796919	The Lost Hero	0	book open jason awaken school bu unabl rememb ...
2	3910776	The Eyes of the Overworld	0	cugel easili persuad merchant fianosth attempt...
3	5969644	Magic's Promise	0	book open heraldmag vanyel return countri vald...
4	3173445	Taran Wanderer	0	taran gurgi return caer dallben follow event t...
...	...	...	...	...
2995	10372180	White Death	5	novel numa file kurt austin adventur novel mai...
2996	14504372	Venus with Pistol	5	gilbert kemp dealer special antiqu gun london ...
2997	3617412	Blackwater	5	know your deep davey alway live shadow older b...
2998	11320975	The Rainbow and the Rose	5	stori concern life johnni pasco retir commerci...
2999	17227674	Chiefs	5	first chief henri lee novel open grow town del...

3000 rows × 4 columns

In [15]:
```
count_vec = CountVectorizer(max_df=0.90,min_df=2,
                            max_features=1000,stop_words='english')
bagofword_vec = count_vec.fit_transform(books['summary'])
bagofword_vec
test = books['genre']
X_train, X_test, y_train, y_test = train_test_split(bagofword_vec,test,test_size=0.2)
X_train.shape,X_test.shape
```

Out[15]:
```
((2400, 1000), (600, 1000))
```

In [16]:
```
%%time

import torch
from transformers import RobertaTokenizer, RobertaForSequenceClassification
from torch.utils.data import Dataset, DataLoader
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
import numpy as np
from tqdm import tqdm

# Custom Dataset class
class BookDataset(Dataset):
    def __init__(self, texts, labels, tokenizer, max_length=512):
        self.texts = texts
        self.labels = labels
        self.tokenizer = tokenizer
        self.max_length = max_length

    def __len__(self):
        return len(self.texts)

    def __getitem__(self, idx):
        text = str(self.texts[idx])
        label = self.labels[idx]

        encoding = self.tokenizer.encode_plus(
            text,
```

```
            add_special_tokens=True,
            max_length=self.max_length,
            padding='max_length',
            truncation=True,
            return_attention_mask=True,
            return_tensors='pt'
        )

        return {
            'input_ids': encoding['input_ids'].flatten(),
            'attention_mask': encoding['attention_mask'].flatten(),
            'labels': torch.tensor(label, dtype=torch.long)
        }

def train_epoch(model, data_loader, optimizer, device):
    model.train()
    total_loss = 0

    for batch in tqdm(data_loader, desc="Training"):
        input_ids = batch['input_ids'].to(device)
        attention_mask = batch['attention_mask'].to(device)
        labels = batch['labels'].to(device)

        optimizer.zero_grad()
        outputs = model(input_ids=input_ids, attention_mask=attention_mask, labels=labels)
        loss = outputs.loss
        total_loss += loss.item()

        loss.backward()
        optimizer.step()

    return total_loss / len(data_loader)

def evaluate(model, data_loader, device):
    model.eval()
    predictions = []
    actual_labels = []

    with torch.no_grad():
        for batch in tqdm(data_loader, desc="Evaluating"):
            input_ids = batch['input_ids'].to(device)
            attention_mask = batch['attention_mask'].to(device)
            labels = batch['labels']

            outputs = model(input_ids=input_ids, attention_mask=attention_mask)
            _, preds = torch.max(outputs.logits, dim=1)

            predictions.extend(preds.cpu().numpy())
            actual_labels.extend(labels.cpu().numpy())

    return accuracy_score(actual_labels, predictions)

# Main execution
def train_roberta_classifier(books, num_epochs=3, batch_size=16):
    # Prepare data
    X_train, X_test, y_train, y_test = train_test_split(
        books['summary'].values,
        books['genre'].values,
        test_size=0.2,
        random_state=42
    )

    # Initialize tokenizer and encode labels
    tokenizer = RobertaTokenizer.from_pretrained('roberta-base')
    label_encoder = LabelEncoder()
    y_train_encoded = label_encoder.fit_transform(y_train)
    y_test_encoded = label_encoder.transform(y_test)

    # Create datasets
    train_dataset = BookDataset(X_train, y_train_encoded, tokenizer)
    test_dataset = BookDataset(X_test, y_test_encoded, tokenizer)

    train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
    test_loader = DataLoader(test_dataset, batch_size=batch_size)

    # Initialize model
    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
    model = RobertaForSequenceClassification.from_pretrained(
        'roberta-base',
        num_labels=len(label_encoder.classes_)
    ).to(device)

    # Initialize optimizer
    optimizer = torch.optim.AdamW(model.parameters(), lr=2e-5)

    # Training loop
    best_accuracy = 0

    for epoch in range(num_epochs):
```

```
        print(f"\nEpoch {epoch + 1}/{num_epochs}")
        avg_loss = train_epoch(model, train_loader, optimizer, device)
        accuracy = evaluate(model, test_loader, device)

        print(f"Average loss: {avg_loss:.4f}")
        print(f"Accuracy: {accuracy:.4f}")

        if accuracy > best_accuracy:
            best_accuracy = accuracy
            # Optionally save the best model
            # torch.save(model.state_dict(), 'best_model.pth')

    return model, label_encoder, best_accuracy

# Example usage
if __name__ == "__main__":
    # Assuming 'books' DataFrame is already loaded
    model, label_encoder, best_accuracy = train_roberta_classifier(books)
    print(f"\nBest accuracy achieved: {best_accuracy:.4f}")
```

```
Some weights of RobertaForSequenceClassification were not initialized from the model checkpoint at roberta-base
and are newly initialized: ['classifier.dense.bias', 'classifier.dense.weight', 'classifier.out_proj.bias', 'cl
assifier.out_proj.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
Epoch 1/3
Training: 100%|██████████| 150/150 [03:36<00:00,  1.44s/it]
Evaluating: 100%|██████████| 38/38 [00:19<00:00,  1.91it/s]
Average loss: 1.4216
Accuracy: 0.6533

Epoch 2/3
Training: 100%|██████████| 150/150 [03:46<00:00,  1.51s/it]
Evaluating: 100%|██████████| 38/38 [00:19<00:00,  1.92it/s]
Average loss: 0.9330
Accuracy: 0.6767

Epoch 3/3
Training: 100%|██████████| 150/150 [03:46<00:00,  1.51s/it]
Evaluating: 100%|██████████| 38/38 [00:19<00:00,  1.91it/s]
Average loss: 0.7305
Accuracy: 0.6833

Best accuracy achieved: 0.6833
CPU times: user 12min 12s, sys: 2.18 s, total: 12min 14s
Wall time: 12min 21s
```

In [17]:

```
%%time

import torch
from transformers import RobertaTokenizer
import pandas as pd
import numpy as np

def predict_genre(text, model, tokenizer, label_encoder, device, max_length=512):
    """
    Predict genre for a single text using the trained RoBERTa model
    """
    # Prepare the text
    encoding = tokenizer.encode_plus(
        text,
        add_special_tokens=True,
        max_length=max_length,
        padding='max_length',
        truncation=True,
        return_attention_mask=True,
        return_tensors='pt'
    )

    # Move to device
    input_ids = encoding['input_ids'].to(device)
    attention_mask = encoding['attention_mask'].to(device)

    # Get prediction
    model.eval()
    with torch.no_grad():
        outputs = model(input_ids=input_ids, attention_mask=attention_mask)
        _, prediction = torch.max(outputs.logits, dim=1)

    # Convert prediction to genre label
    predicted_genre = label_encoder.inverse_transform([prediction.item()])[0]
    return predicted_genre
```

```python
def predict_genres_batch(texts, model, tokenizer, label_encoder, device, batch_size=32, max_length=512):
    """
    Predict genres for a batch of texts
    """
    model.eval()
    predictions = []

    for i in range(0, len(texts), batch_size):
        batch_texts = texts[i:i + batch_size]

        # Prepare batch
        encodings = tokenizer(
            batch_texts.tolist(),
            add_special_tokens=True,
            max_length=max_length,
            padding='max_length',
            truncation=True,
            return_attention_mask=True,
            return_tensors='pt'
        )

        # Move to device
        input_ids = encodings['input_ids'].to(device)
        attention_mask = encodings['attention_mask'].to(device)

        # Get predictions
        with torch.no_grad():
            outputs = model(input_ids=input_ids, attention_mask=attention_mask)
            _, batch_predictions = torch.max(outputs.logits, dim=1)

        # Convert predictions to genre labels
        batch_genres = label_encoder.inverse_transform(batch_predictions.cpu().numpy())
        predictions.extend(batch_genres)

    return predictions

def evaluate_model_performance(books, model, tokenizer, label_encoder):
    """
    Evaluate model performance and add predictions to the books DataFrame
    """
    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

    # Get predictions for all books
    predicted_genres = predict_genres_batch(
        books['summary'],
        model,
        tokenizer,
        label_encoder,
        device
    )

    # Add predictions to DataFrame
    books_with_predictions = books.copy()
    books_with_predictions['predicted_genre'] = predicted_genres

    # Calculate accuracy
    accuracy = (books_with_predictions['genre'] == books_with_predictions['predicted_genre']).mean()

    return books_with_predictions, accuracy

# Example usage
def main():
    # Assuming we have our trained model, label_encoder from the previous code
    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
    tokenizer = RobertaTokenizer.from_pretrained('roberta-base')

    # Load your books DataFrame
    # books = pd.read_csv('your_books_data.csv')  # Uncomment and modify as needed

    # Evaluate model and get predictions
    books_with_predictions, accuracy = evaluate_model_performance(
        books,
        model,  # Your trained model from previous code
        tokenizer,
        label_encoder  # Your label encoder from previous code
    )

    # Display results
    print(f"\nModel Accuracy: {accuracy:.4f}")
    print("\nSample Predictions:")
    print(books_with_predictions[['genre', 'predicted_genre', 'summary']].head())

    # Optionally save results
    # books_with_predictions.to_csv('predictions_results.csv', index=False)

    return books_with_predictions

if __name__ == "__main__":
    books_with_predictions = main()
```

```
Model Accuracy: 0.8160

Sample Predictions:
   genre  predicted_genre                                            summary
0      0                0  drown wednesday first truste among morrow day ...
1      0                0  book open jason awaken school bu unabl rememb ...
2      0                0  cugel easili persuad merchant fianosth attempt...
3      0                0  book open heraldmag vanyel return countri vald...
4      0                0  taran gurgi return caer dallben follow event t...
CPU times: user 1min 38s, sys: 45.9 ms, total: 1min 38s
Wall time: 1min 38s
```

In [18]:
```
print("\nSample Predictions:")
books_with_predictions[['genre', 'predicted_genre', 'summary']].head()
```

```
Sample Predictions:
```

Out[18]:

	genre	predicted_genre	summary
0	0	0	drown wednesday first truste among morrow day ...
1	0	0	book open jason awaken school bu unabl rememb ...
2	0	0	cugel easili persuad merchant fianosth attempt...
3	0	0	book open heraldmag vanyel return countri vald...
4	0	0	taran gurgi return caer dallben follow event t...

In [19]:
```
books_with_predictions[['genre', 'predicted_genre', 'summary']].sample(10)
```

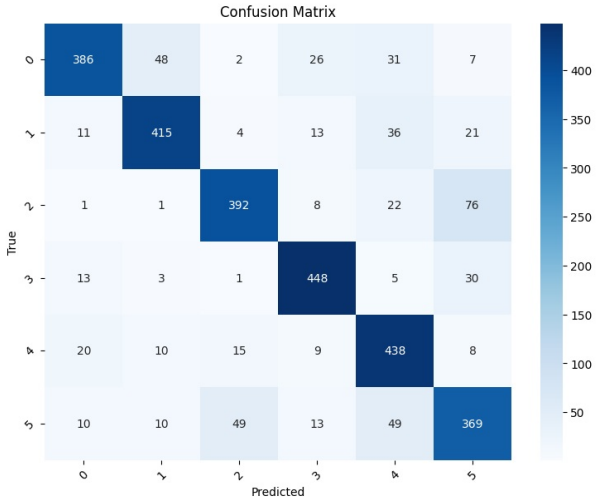
Out[19]:

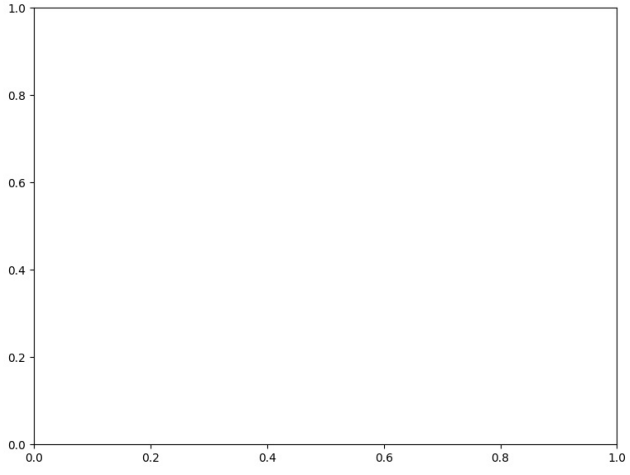
	genre	predicted_genre	summary
1535	3	3	midth centuri father gstir sent bavaria canada...
1317	2	2	week rebuss retir rebu clark investig death fa...
414	0	0	amurtaht black dragon exist plagu fief heltant...
1812	3	5	tale pick month conclus david liss first novel...
19	0	0	six month took control territori crimin organ ...
927	1	1	cassi come home owl morph find horkbajir leav ...
2484	4	4	golden reveal thing plotlin interview slayerli...
1716	3	3	beefsteak club lord john grey introduc robert ...
2441	4	1	novel take place event umbrella conspiraci res...
900	1	1	presentday megatron lie dead bottom sea cut gu...

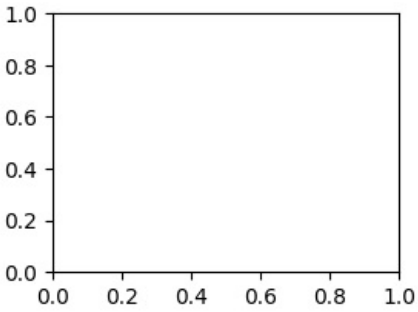
In [20]:
```
%%time

import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix, classification_report, roc_curve, auc
from sklearn.preprocessing import label_binarize
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

def plot_metrics(books_with_predictions, label_encoder, model, tokenizer, device):
    """
    Plot comprehensive metrics for model evaluation
    """
    true_labels = books_with_predictions['genre']
    predicted_labels = books_with_predictions['predicted_genre']

    # Set up the plotting style

    # Create a figure with subplots
    fig = plt.figure(figsize=(20, 15))

    # 1. Confusion Matrix
    plt.subplot(2, 2, 1)
    cm = confusion_matrix(true_labels, predicted_labels)
    sns.heatmap(cm,
                annot=True,
                fmt='d',
                cmap='Blues',
                xticklabels=label_encoder.classes_,
                yticklabels=label_encoder.classes_)
    plt.title('Confusion Matrix')
    plt.xlabel('Predicted')
    plt.ylabel('True')
    plt.xticks(rotation=45)
    plt.yticks(rotation=45)
```

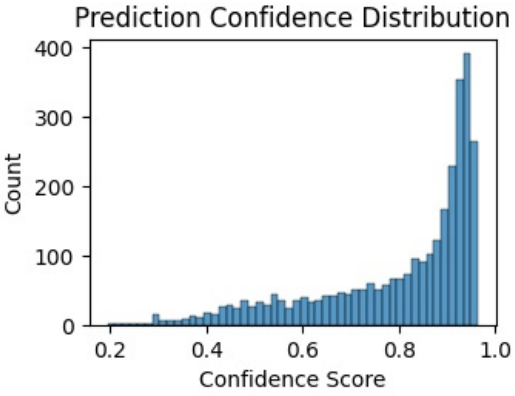
```
    # 2. Genre Distribution
    plt.subplot(2, 2, 2)
    genre_counts = pd.DataFrame({
        'True': true_labels.value_counts(),
        'Predicted': pd.Series(predicted_labels).value_counts()
    }).fillna(0)
    genre_counts.plot(kind='bar')
    plt.title('Genre Distribution: True vs Predicted')
    plt.xlabel('Genre')
    plt.ylabel('Count')
    plt.xticks(rotation=45)
    plt.legend()

    # 3. Per-Class Accuracy
    plt.subplot(2, 2, 3)
    class_accuracy = pd.DataFrame({
        'Accuracy': classification_report(true_labels, predicted_labels, output_dict=True)['weighted avg']
    }, index=['Precision', 'Recall', 'F1-Score'])
    class_accuracy.plot(kind='bar')
    plt.title('Overall Model Metrics')
    plt.ylabel('Score')
    plt.ylim(0, 1)

    # 4. Prediction Confidence Distribution
    plt.subplot(2, 2, 4)

    # Get prediction probabilities
    def get_prediction_probabilities(texts):
        model.eval()
        probs = []

        with torch.no_grad():
            for text in texts:
                encoding = tokenizer.encode_plus(
                    text,
                    add_special_tokens=True,
                    max_length=512,
                    padding='max_length',
                    truncation=True,
                    return_attention_mask=True,
                    return_tensors='pt'
                )

                input_ids = encoding['input_ids'].to(device)
                attention_mask = encoding['attention_mask'].to(device)

                outputs = model(input_ids=input_ids, attention_mask=attention_mask)
                prob = torch.softmax(outputs.logits, dim=1).max(dim=1)[0]
                probs.append(prob.cpu().item())

        return probs

    prediction_probs = get_prediction_probabilities(books_with_predictions['summary'])
    sns.histplot(prediction_probs, bins=50)
    plt.title('Prediction Confidence Distribution')
    plt.xlabel('Confidence Score')
    plt.ylabel('Count')

    plt.tight_layout()
    plt.show()

    # Print detailed classification report
    print("\nClassification Report:")
    print(classification_report(true_labels, predicted_labels))

    # Plot ROC curves for each class
    plt.figure(figsize=(10, 8))

    # Prepare data for ROC curves
    y_test_bin = label_binarize(true_labels, classes=label_encoder.classes_)
    n_classes = len(label_encoder.classes_)

    # Get prediction probabilities for all classes
    def get_all_probabilities(texts):
        model.eval()
        all_probs = []

        with torch.no_grad():
            for text in texts:
                encoding = tokenizer.encode_plus(
                    text,
                    add_special_tokens=True,
                    max_length=512,
                    padding='max_length',
                    truncation=True,
                    return_attention_mask=True,
                    return_tensors='pt'
                )
```

```
                input_ids = encoding['input_ids'].to(device)
                attention_mask = encoding['attention_mask'].to(device)

                outputs = model(input_ids=input_ids, attention_mask=attention_mask)
                probs = torch.softmax(outputs.logits, dim=1)
                all_probs.append(probs.cpu().numpy())

        return np.vstack(all_probs)

    y_score = get_all_probabilities(books_with_predictions['summary'])

    # Plot ROC curve for each class
    for i in range(n_classes):
        fpr, tpr, _ = roc_curve(y_test_bin[:, i], y_score[:, i])
        roc_auc = auc(fpr, tpr)
        plt.plot(fpr, tpr, label=f'{label_encoder.classes_[i]} (AUC = {roc_auc:.2f})')

    plt.plot([0, 1], [0, 1], 'k--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('ROC Curves for Each Genre')
    plt.legend(loc="lower right", bbox_to_anchor=(1.7, 0.5))
    plt.show()

    # Calculate and display additional metrics
    correct_predictions = sum(true_labels == predicted_labels)
    total_predictions = len(true_labels)
    accuracy = correct_predictions / total_predictions

    print(f"\nAdditional Metrics:")
    print(f"Total Samples: {total_predictions}")
    print(f"Correct Predictions: {correct_predictions}")
    print(f"Overall Accuracy: {accuracy:.4f}")

    # Per-genre analysis
    print("\nPer-Genre Performance:")
    per_genre_accuracy = {}
    for genre in label_encoder.classes_:
        genre_mask = true_labels == genre
        genre_total = sum(genre_mask)
        genre_correct = sum((true_labels == predicted_labels) & genre_mask)
        genre_accuracy = genre_correct / genre_total if genre_total > 0 else 0
        per_genre_accuracy[genre] = genre_accuracy
        print(f"{genre}:")
        print(f"  Total samples: {genre_total}")
        print(f"  Correct predictions: {genre_correct}")
        print(f"  Accuracy: {genre_accuracy:.4f}")

    return per_genre_accuracy

def visualize_model_performance(books_with_predictions, model, label_encoder):
    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
    tokenizer = RobertaTokenizer.from_pretrained('roberta-base') # Initialize tokenizer here
    per_genre_accuracy = plot_metrics(books_with_predictions, label_encoder, model, tokenizer, device) # Pass t
    return per_genre_accuracy


if __name__ == "__main__":
    # ... your existing code ...

    visualize_model_performance(books_with_predictions, model, label_encoder)
```



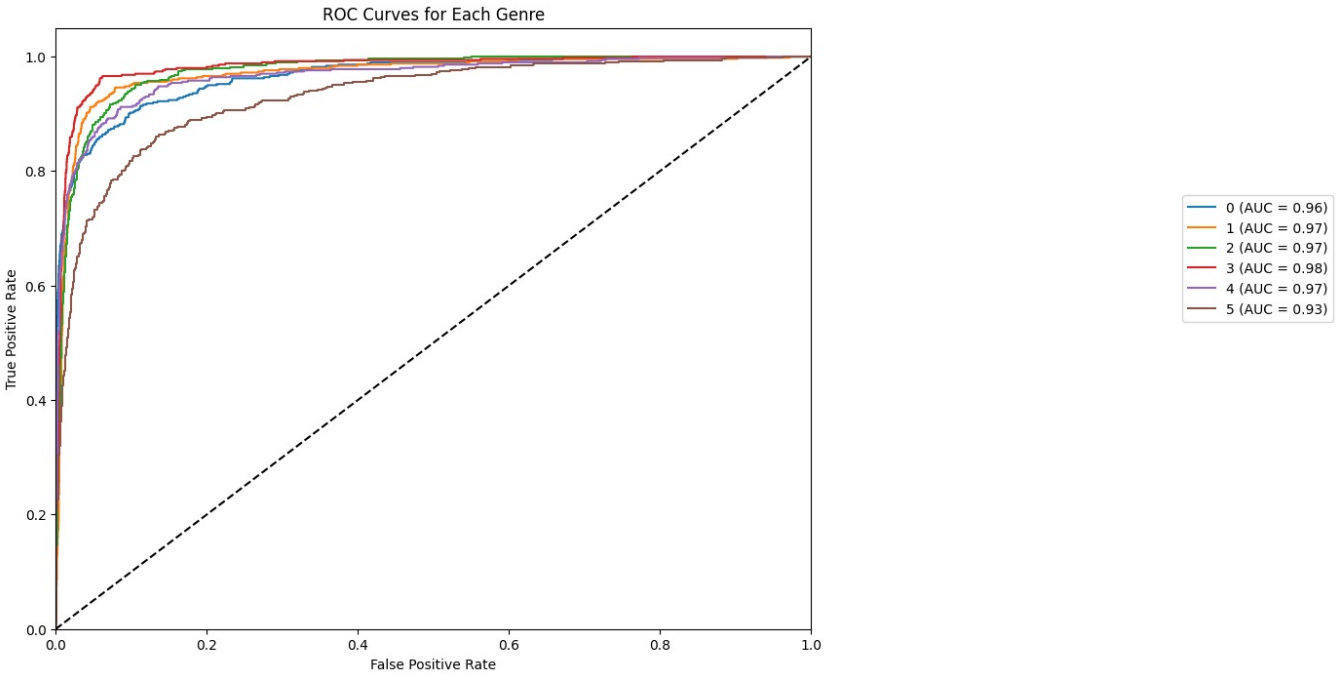






Classification Report:

	precision	recall	f1-score	support
0	0.88	0.77	0.82	500
1	0.85	0.83	0.84	500
2	0.85	0.78	0.81	500
3	0.87	0.90	0.88	500
4	0.75	0.88	0.81	500
5	0.72	0.74	0.73	500
accuracy			0.82	3000
macro avg	0.82	0.82	0.82	3000
weighted avg	0.82	0.82	0.82	3000



```
Additional Metrics:
Total Samples: 3000
Correct Predictions: 2448
Overall Accuracy: 0.8160

Per-Genre Performance:
0:
  Total samples: 500
  Correct predictions: 386
  Accuracy: 0.7720
1:

  Total samples: 500
  Correct predictions: 415
  Accuracy: 0.8300
2:
  Total samples: 500
  Correct predictions: 392
  Accuracy: 0.7840
3:
  Total samples: 500
  Correct predictions: 448
  Accuracy: 0.8960
4:
  Total samples: 500
  Correct predictions: 438
  Accuracy: 0.8760
5:
  Total samples: 500
  Correct predictions: 369
  Accuracy: 0.7380
CPU times: user 3min 24s, sys: 926 ms, total: 3min 25s
Wall time: 3min 23s
```

In [21]:

```python
import torch
from transformers import RobertaTokenizer, RobertaForSequenceClassification

def predict_book_genre(model, label_encoder, tokenizer, text):
    """
    Predict the genre of a given book summary text.

    Parameters:
    model (RobertaForSequenceClassification): The trained RoBERTa classification model.
    label_encoder (LabelEncoder): The label encoder used to transform the labels.
    tokenizer (RobertaTokenizer): The tokenizer used to preprocess the input text.
    text (str): The book summary text to classify.

    Returns:
    str: The predicted genre of the book.
    """
    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

    # Encode the input text
    encoding = tokenizer.encode_plus(
        text,
        add_special_tokens=True,
        max_length=512,
        padding='max_length',
        truncation=True,
        return_attention_mask=True,
        return_tensors='pt'
    )

    input_ids = encoding['input_ids'].to(device)
    attention_mask = encoding['attention_mask'].to(device)

    # Make the prediction
    with torch.no_grad():
        outputs = model(input_ids=input_ids, attention_mask=attention_mask)
        _, predicted_idx = torch.max(outputs.logits, dim=1)

    # Decode the predicted label
    predicted_label = label_encoder.inverse_transform([predicted_idx.item()])[0]

    return predicted_label

# Example usage
if __name__ == "__main__":
    # Assuming you have the trained model, label encoder, and a book summary
    book_summary = "This is a captivating story about a young wizard who discovers a magical world hidden from

    # Load the trained model and tokenizer
    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
    model = RobertaForSequenceClassification.from_pretrained('roberta-base', num_labels=len(label_encoder.class
    tokenizer = RobertaTokenizer.from_pretrained('roberta-base')

    prediction = predict_book_genre(model, label_encoder, tokenizer, book_summary)
    print(f"Predicted genre: {prediction}")
```

```
Some weights of RobertaForSequenceClassification were not initialized from the model checkpoint at roberta-base
and are newly initialized: ['classifier.dense.bias', 'classifier.dense.weight', 'classifier.out_proj.bias', 'cl
assifier.out_proj.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
Predicted genre: 0
```