In [1]:
```
!kaggle datasets download jacopoferretti/bbc-articles-dataset
!unzip bbc-articles-dataset.zip
```

```
Dataset URL: https://www.kaggle.com/datasets/jacopoferretti/bbc-articles-dataset (https://www.kaggle.com/datasets/jacopoferretti/bbc-articles-dataset)
License(s): CC0-1.0
Downloading bbc-articles-dataset.zip to /content
  0% 0.00/1.81M [00:00<?, ?B/s]
100% 1.81M/1.81M [00:00<00:00, 130MB/s]
Archive:  bbc-articles-dataset.zip
  inflating: bbc_text_cls.csv
```

In [2]:
```
import pandas as pd
```

In [3]:
```
df = pd.read_csv("/content/bbc_text_cls.csv")
```

In [4]:
```
df.head()
```

Out[4]:

	text	labels
0	Ad sales boost Time Warner profit\n\nQuarterly...	business
1	Dollar gains on Greenspan speech\n\nThe dollar...	business
2	Yukos unit buyer faces loan claim\n\nThe owner...	business
3	High fuel prices hit BA's profits\n\nBritish A...	business
4	Pernod takeover talk lifts Domecq\n\nShares in...	business

Pre-Processing Text

In [5]:
```
import re
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from bs4 import BeautifulSoup
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
```

Out[5]: True

In [6]:

```python
def preprocess_text(text):
    # 1. Lowercasing
    text = text.lower()

    # 2. Remove HTML Tags
    text = BeautifulSoup(text, 'html.parser').get_text()

    # 3. Remove URLs
    text = re.sub(r'https?://\S+|www\.\S+', '', text)

    # 4. Remove emojis
    emoji_pattern = re.compile("["
                               "\U0001F600-\U0001F64F"  # Emoti
                               "\U0001F300-\U0001F5FF"  # Misce
                               "\U0001F680-\U0001F6FF"  # Trans
                               "\U0001F700-\U0001F77F"  # Alche
                               "\U0001F780-\U0001F7FF"  # Geome
                               "\U0001F800-\U0001F8FF"  # Suppl
                               "\U0001F900-\U0001F9FF"  # Suppl
                               "\U0001FA00-\U0001FA6F"  # Chess
                               "\U0001FA70-\U0001FAFF"  # Symbo
                               "\U00002702-\U000027B0"  # Dingb
                               "\U000024C2-\U0001F251"
                               "]+", flags=re.UNICODE)
    text = emoji_pattern.sub('', text)

    # 5. Remove punctuation and numbers
    text = re.sub(r'[^a-z\s]', '', text)

    # 6. Tokenization
    tokens = word_tokenize(text)

    # 7. Remove stopwords and single character tokens
    stop_words = set(stopwords.words('english'))
    tokens = [token for token in tokens if token not in stop_wo

    # 8. Lemmatization
    lemmatizer = WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(token) for token in tokens]

    # Joining tokens back into a sentence
    preprocessed_text = ' '.join(tokens)

    return preprocessed_text
```

In [7]:

```python
df['text'] = df['text'].apply(preprocess_text)
```

In [8]:
```
df
```

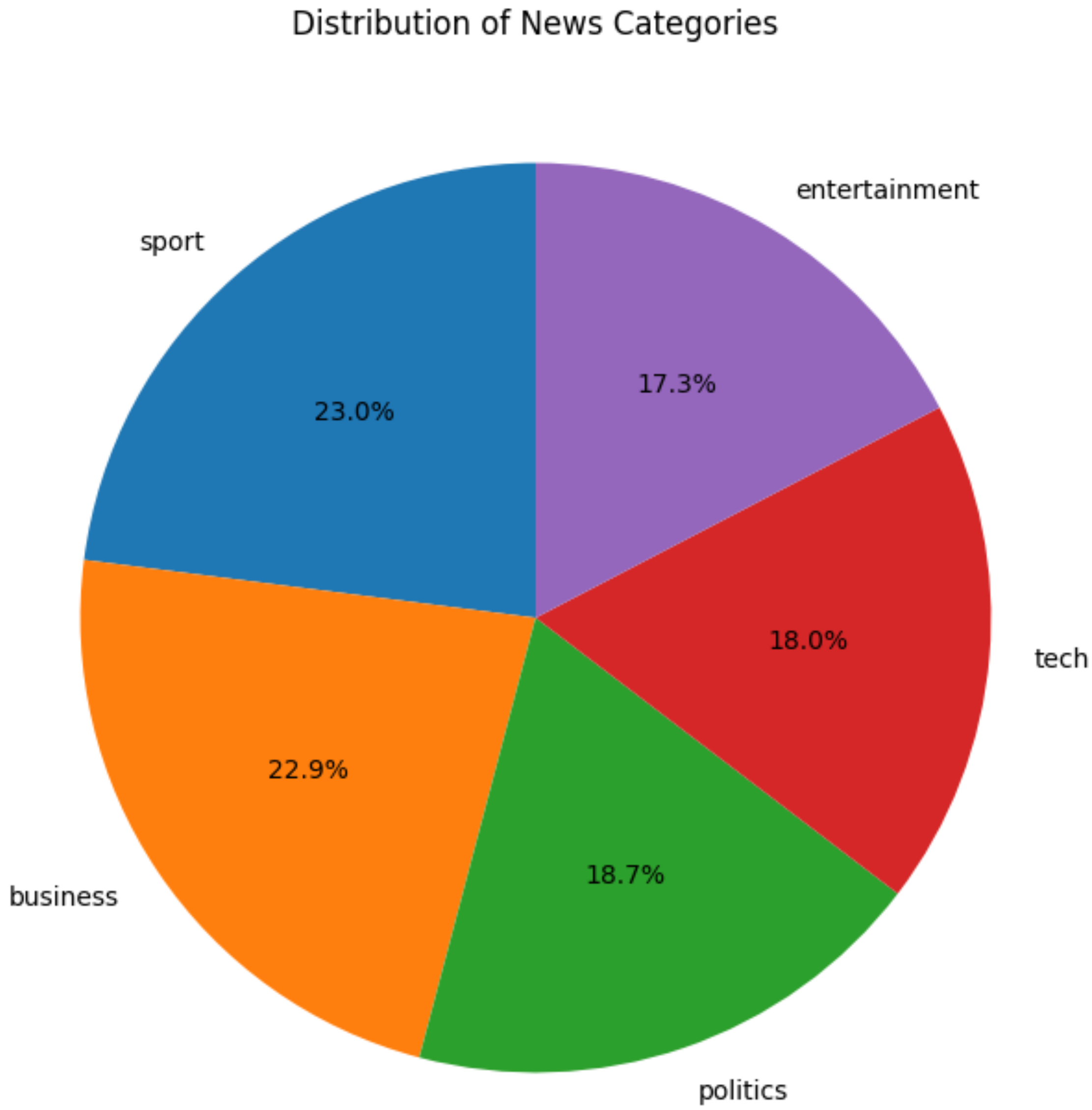
Out[8]:

	text	labels
0	ad sale boost time warner profit quarterly pro...	business
1	dollar gain greenspan speech dollar hit highes...	business
2	yukos unit buyer face loan claim owner embattl...	business
3	high fuel price hit ba profit british airway b...	business
4	pernod takeover talk lift domecq share uk drin...	business
...	...	...
2220	bt program beat dialler scam bt introducing tw...	tech
2221	spam email tempt net shopper computer user acr...	tech
2222	careful code new european directive could put ...	tech
2223	u cyber security chief resigns man making sure...	tech
2224	losing online gaming online role playing game ...	tech

2225 rows × 2 columns

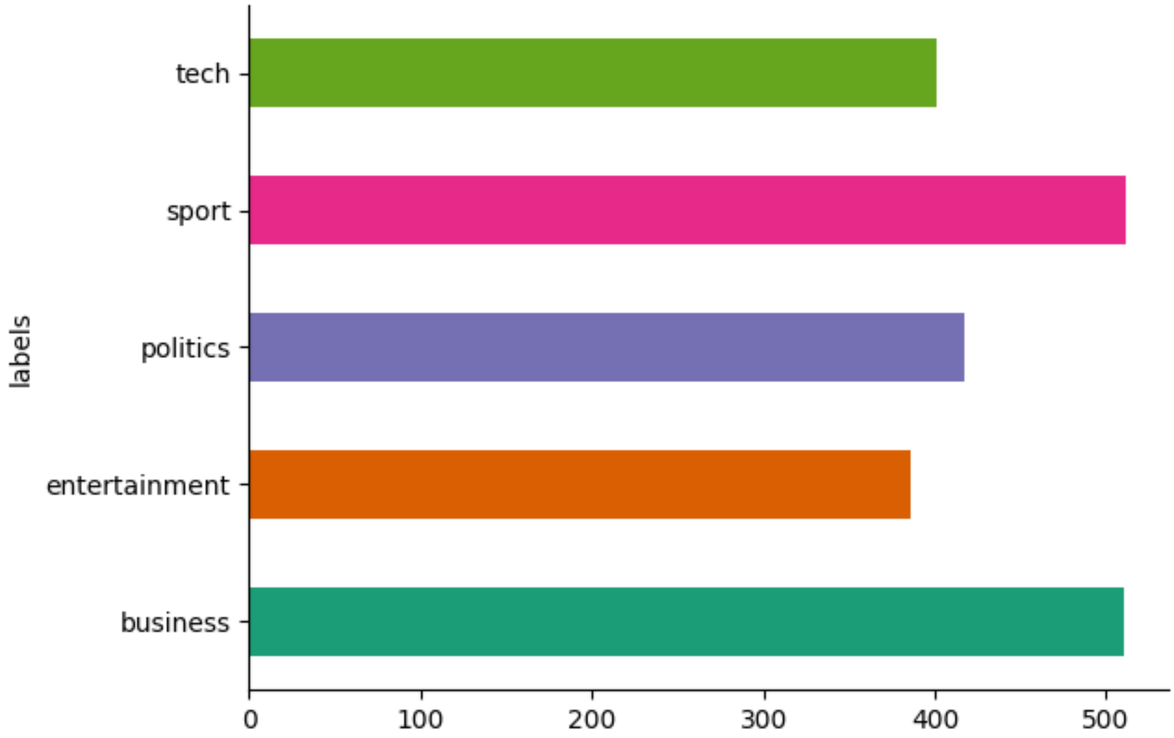
In [9]:

```
# @title Distribution of News Categories

import matplotlib.pyplot as plt

category_counts = df['labels'].value_counts()

plt.figure(figsize=(8, 8))
plt.pie(category_counts, labels=category_counts.index, autopct=
_ = plt.title('Distribution of News Categories')
```



In [10]:

```
# @title labels

from matplotlib import pyplot as plt
import seaborn as sns
df.groupby('labels').size().plot(kind='barh', color=sns.palette
plt.gca().spines[['top', 'right',]].set_visible(False)
```



Classification Using Machine learnig model

Convert Text to Numerical Format (Tokenization)

---

1. Bag-of-Words (BoW)

- Counts the occurrence of each word in a document.
- Generates sparse, high-dimensional vectors where each word is a feature.
- Pros: Simple and effective for many applications.
- Cons: Ignores word order and meaning.
- **Implementation**: `CountVectorizer` in Scikit-Learn.

---

2. Term Frequency-Inverse Document Frequency (TF-IDF)

- Weighs word importance based on how frequently it appears in a document vs. across all documents.
- Useful for emphasizing significant words that appear frequently in a document but

not commonly across documents.
- Pros: Reduces the impact of common but less informative words.
- Cons: Still ignores word order and context.
- **Implementation**: `TfidfVectorizer` in Scikit-Learn.

---

3. n-grams

- Extracts contiguous sequences of `n` words to capture basic word order and simple context.
- Common choices: bigrams (n=2), trigrams (n=3).
- Pros: Adds some contextual information beyond individual words.
- Cons: Increases feature space and can add noise if `n` is too large.
- **Implementation**: `CountVectorizer` or `TfidfVectorizer` with `ngram_range` parameter in Scikit-Learn.

---

4. Word Embeddings

- Maps words to dense, low-dimensional vectors that capture semantic meaning.
- Popular pre-trained embeddings:
  - **Word2Vec**: Trained on word co-occurrences; represents semantic relationships.
  - **GloVe**: Trained on global word-word co-occurrence statistics.
- Pros: Captures meaning and relationships between words.
- Cons: Requires pre-trained embeddings and doesn't inherently capture sentence structure.
- **Implementation**: Using libraries like `gensim` or loading pre-trained embeddings in PyTorch/Keras.

---

5. Pre-trained Transformer Embeddings

- Use pre-trained models like BERT, DistilBERT, or RoBERTa to extract embeddings for text.
- Contextualized embeddings capture word meaning based on context.
- Pros: State-of-the-art embeddings for most NLP tasks; captures contextual meaning.
- Cons: Requires more computation and memory.
- **Implementation**: `Hugging Face Transformers` library.

In [41]:
```
# @title Spliting Data

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(df['text'],
```

In [53]:

```
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)
y_test_encoded = label_encoder.transform(y_test)
```

In [43]:

```
# @title 1. Bag-of-Words (BoW)

from sklearn.feature_extraction.text import CountVectorizer

bow_vectorizer = CountVectorizer()

X_train_bow = bow_vectorizer.fit_transform(X_train)
X_test_bow = bow_vectorizer.transform(X_test)
```

In [58]:

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_repo

nb_bow = MultinomialNB()
nb_bow.fit(X_train_bow, y_train)

# Predict on the test data
y_pred_bow = nb_bow.predict(X_test_bow)

# Evaluate the BoW model
print("Bag-of-Words Model Evaluation")
print("Accuracy:", accuracy_score(y_test, y_pred_bow))
print("Classification Report:\n", classification_report(y_test,
```

```
Bag-of-Words Model Evaluation
Accuracy: 0.9685393258426966
Classification Report:
                precision    recall  f1-score   support

     business       0.97      0.96      0.96       115
entertainment       0.99      0.93      0.96        72
     politics       0.93      0.97      0.95        76
        sport       1.00      0.99      1.00       102
         tech       0.95      0.99      0.97        80

     accuracy                           0.97       445
    macro avg       0.97      0.97      0.97       445
 weighted avg       0.97      0.97      0.97       445
```

In [49]:

```
# @title 2. Term Frequency-Inverse Document Frequency (TF-IDF)
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import accuracy_score, classification_repo

tfidf_vectorizer = TfidfVectorizer()

# Transform the text data into TF-IDF features
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)

# Train a Naive Bayes classifier on TF-IDF features
nb_tfidf = MultinomialNB()
nb_tfidf.fit(X_train_tfidf, y_train)

# Predict on the test data
y_pred_tfidf = nb_tfidf.predict(X_test_tfidf)

# Evaluate the TF-IDF model
print("TF-IDF Model Evaluation")
print("Accuracy:", accuracy_score(y_test, y_pred_tfidf))
print("Classification Report:\n", classification_report(y_test,
```

```
TF-IDF Model Evaluation
Accuracy: 0.9617977528089887
Classification Report:
                precision    recall  f1-score   support

     business       0.97      0.95      0.96       115
entertainment       0.98      0.90      0.94        72
     politics       0.90      0.97      0.94        76
        sport       0.99      0.99      0.99       102
         tech       0.95      0.99      0.97        80

     accuracy                           0.96       445
    macro avg       0.96      0.96      0.96       445
 weighted avg       0.96      0.96      0.96       445
```

In [50]:

```
text = '''
wru proposes season overhaul welsh rugby union want restructure
start celtic league october followed heineken cup february march
twomonth period away home international match wru chairman david
club country added feel sure spectator interest would respond im
timetable currently operation equally suspect sponsor would pref
would also enjoy increased exposure moving six nation traditiona
greater interest game generally provide increased skill competit
international rugby board next month four plan drawn independent
early day number caveat associated least revenue broadcaster ext
'''
```

In [67]:

```
# @title Tfidf Prediction

nb_tfidf.predict(tfidf_vectorizer.transform([text]))[0]
```

Out[67]: 'sport'

In [68]:

```
# @title BOW Prediction

nb_bow.predict(bow_vectorizer.transform([text]))[0]
```

Out[68]: 'sport'