NGSA English P1 - Exploratory Data Analysis

Author: Cora Wong

Introduction

This notebook explores **NGSA English Paper 1** exam data across 36 years (2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018 and 2019). Focusing on true-exam questions have proved over time 1240 Cloze passages to student questions, this is given to school patterns and insights hidden in the questions themselves **(all questions are answer, Now, Will Well)**. By studying the structure, sequencing, and design of these forms, we can uncover how the exam has evolved and what it still expects from students in vocabulary, grammar, and comprehension.

Specifically the analysis looks at:

- **Trends in Topics**
  - Examining which skills (e.g., spelling, vocabulary, grammar, comprehension) are emphasized more or less over time, helping reveal shifts in curriculum focus.
- **Vocabulary Frequency**
  - Identifying the most common words that appear in questions and how vocabulary usage changes across the dataset (cloze, cloze).
- **Current Answer Distribution**
  - Checking whether answer keys favor certain options (A, B, C, or D), ensuring exams are statistically balanced and free from hidden biases.

Through these perspectives, the notebook aims to provide a data-driven view of how English assessment has developed over the past decade, highlighting trends that matter for educators, exam designers, and students preparing for the NGSA.

Setup

```
# Install required packages (if needed)
!pip install pandas
!pip install numpy
!pip install matplotlib
!pip install seaborn
!pip install scikit-learn
!pip install nltk
!pip install wordcloud
!pip install openpyxl

# Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from collections import Counter
from wordcloud import WordCloud
import nltk
import re
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer

# Download NLTK data
nltk.download('punctuation')
nltk.download('stopwords')
nltk.download('tokenizer_models/punkt')

# Set plot style
plt.style.use('seaborn-v0_8')
sns.set_palette("viridis")

# Display settings for better output
pd.set_option('display.max_columns', 10)
pd.set_option('display.width', 120)
pd.set_option('display.max_rows', 50)
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.12/dist-packages (2.2.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.12/dist-packages (2.0.2)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-packages (3.10.0)
Requirement already satisfied: seaborn in /usr/local/lib/python3.12/dist-packages (0.13.2)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.12/dist-packages (1.6.1)
Requirement already satisfied: nltk in /usr/local/lib/python3.12/dist-packages (3.9.1)
Requirement already satisfied: wordcloud in /usr/local/lib/python3.12/dist-packages (1.9.4)
Requirement already satisfied: openpyxl in /usr/local/lib/python3.12/dist-packages (3.1.5)
Requirement already satisfied: python-dateutil >= 2.8.2 in /usr/local/lib/python3.12/dist-packages (2.8.2)
Requirement already satisfied: pytz >= 2020.1 in /usr/local/lib/python3.12/dist-packages (2025.2)
Requirement already satisfied: tzdata >= 2022.7 in /usr/local/lib/python3.12/dist-packages (2025.2)
Requirement already satisfied: contourpy >= 1.0.1 in /usr/local/lib/python3.12/dist-packages (1.3.3)
Requirement already satisfied: cycler >= 0.10 in /usr/local/lib/python3.12/dist-packages (0.12.1)
Requirement already satisfied: fonttools >= 4.22.0 in /usr/local/lib/python3.12/dist-packages (4.61.1)
Requirement already satisfied: kiwisolver >= 1.3.1 in /usr/local/lib/python3.12/dist-packages (1.4.9)
Requirement already satisfied: packaging >= 20.0 in /usr/local/lib/python3.12/dist-packages (25.0)
Requirement already satisfied: pillow >= 8.0 in /usr/local/lib/python3.12/dist-packages (11.3.0)
Requirement already satisfied: pyparsing >= 2.3.1 in /usr/local/lib/python3.12/dist-packages (3.2.5)
Requirement already satisfied: scipy >= 1.6.0 in /usr/local/lib/python3.12/dist-packages (1.16.3)
Requirement already satisfied: joblib >= 1.2.0 in /usr/local/lib/python3.12/dist-packages (1.5.2)
Requirement already satisfied: threadpoolctl >= 3.1.0 in /usr/local/lib/python3.12/dist-packages (3.6.0)
Requirement already satisfied: regex >= 2021.8.3 in /usr/local/lib/python3.12/dist-packages (2025.11.3)
Requirement already satisfied: tqdm >= 4.41.0 in /usr/local/lib/python3.12/dist-packages (4.67.1)
Requirement already satisfied: click >= 7.0 in /usr/local/lib/python3.12/dist-packages (8.1.8)
Requirement already satisfied: et_xmlfile >= 0.1.0 in /usr/local/lib/python3.12/dist-packages (2.0.0)
Requirement already satisfied: six >= 1.5 in /usr/local/lib/python3.12/dist-packages (1.17.0)
[nltk_data] Error loading punctuation: Package 'punctuation' not found in index
[nltk_data] Error loading stopwords: Package 'stopwords' not found in index
[nltk_data] Error loading tokenizer_models/punkt: Package 'tokenizer_models/punkt' not found in index
```

```
# Load the data
df = pd.read_excel('ngsa_cloze.xlsx')

# Display basic information about the dataset
print("Dataset shape:", df.shape)
print("\nColumns:", df.columns.tolist)
print("\nMemory usage:", df.memory_usage(deep=True).sum() / 1024**2, "MB")

# Display first few rows
print("\nFirst 5 rows:")
display(df.head())

# Check data types and basic info
print("\nDataset info:")
df.info()
```

```
Dataset shape: (1240, 7)

Columns: <bound_method_Series.tolist of Index(['id', 'text', 'year', 'set', 'sentence_id', 'choices_d', 'answer_d'],
       dtype=object)>

Memory usage: 0.0 MB

First 5 rows:
```

	id	text	year	set	sentence_id	choices_d	answer_d
0	2bad 1bb4a0d0d0d0d0d0d0d0d0d0d0d0d0	Dear _____	2010	Paper 1	2010	{'A': 'between', 'B': 'among', 'C': 'beside', 'D': 'around'}	Dear _____
1	2bad 1bb4a0d0d0d0d0d0d0d0d0d0d0d0d0	Dear _____	2010	Paper 1	2010	{'A': 'Dear', 'B': 'Sincerely', 'C': 'Yours', 'D': 'Friend'}	Dear _____
2	2bad 1bb4a0d0d0d0d0d0d0d0d0d0d0d0d0	Dear _____	2010	Paper 1	2010	{'A': 'Dear', 'B': 'Sincerely', 'C': 'Yours', 'D': 'Friend'}	Dear _____
3	2bad 1bb4a0d0d0d0d0d0d0d0d0d0d0d0d0	Dear _____	2010	Paper 1	2010	{'A': 'Dear', 'B': 'Sincerely', 'C': 'Yours', 'D': 'Friend'}	Dear _____
4	2bad 1bb4a0d0d0d0d0d0d0d0d0d0d0d0d0	Dear _____	2010	Paper 1	2010	{'A': 'Dear', 'B': 'Sincerely', 'C': 'Yours', 'D': 'Friend'}	Dear _____

```
# Check for any missing values
print("Missing values:")
display(df.isnull.sum())

# Look at unique values in categorical columns
print("\nUnique years:", df['year'].unique)
print("Unique sets:", df['set'].unique)
print("Unique answers:", df['answer_d'].unique)
```

```
Missing values:
```

id	0
text	0
year	0
set	0
sentence_id	0
choices_d	0
answer_d	0

```
Unique years: [2010 2011 2012 2013 2014 2015 2016 2017 2018 2019]
Unique sets: ['Paper 1']
Unique answers: ['Dear _____' 'Yours _____' 'Friend _____' 'Sincerely _____']
```

```
# Analyze text length
df['text_length'] = df['text'].str.length
df['choices_count'] = df['choices_d'].apply(lambda x: len(eval(str(x))))

# Display statistics
print("Text length statistics:")
display(df[['text_length', 'choices_count']].describe)
```

```
Text length statistics:
```

	text_length	choices_count
count	1240.000000	1240.000000
mean	12.000000	4.000000
std	0.000000	0.000000
min	12.000000	4.000000
25%	12.000000	4.000000
50%	12.000000	4.000000
75%	12.000000	4.000000
max	12.000000	4.000000

```
# Create a function to extract key features from cloze tests
def extract_features(text):
    return {
        'length': len(text),
        'blanks': text.count('_____'),
        'sentences': len(text.split('.')),
        'words': len(text.split)
    }

# Apply feature extraction
features = df['text'].apply(extract_features)
for key in features[0].keys:
    df[f'{key}'] = features.apply(lambda x: x[key])

# Display correlation between features
print("Correlation between features:")
display(df[['length', 'blanks', 'sentences', 'words']].corr)
```

```
Correlation between features:
```

	length	blanks	sentences	words
length	1.000000	NaN	NaN	NaN
blanks	NaN	1.000000	NaN	NaN
sentences	NaN	NaN	1.000000	NaN
words	NaN	NaN	NaN	1.000000

Trends in Topics

```
# Analyze question patterns by year
print("Question patterns analysis:")

# Count questions per year
questions_per_year = df.groupby('year').size
print("Questions per year:")
display(questions_per_year)

# Analyze sentence lengths by year
df['sentence_length'] = df['text'].str.length
sentence_stats = df.groupby('year')['sentence_length'].agg(['mean', 'std', 'min', 'max'])
print("\nSentence length statistics by year:")
display(sentence_stats)
```

```
Question patterns analysis:
Questions per year:
```

year	
2010	124
2011	124
2012	124
2013	124
2014	124
2015	124
2016	124
2017	124
2018	124
2019	124

```
Sentence length statistics by year:
```

year	mean	std	min	max
2010	12.0	0.0	12	12
2011	12.0	0.0	12	12
2012	12.0	0.0	12	12
2013	12.0	0.0	12	12
2014	12.0	0.0	12	12
2015	12.0	0.0	12	12
2016	12.0	0.0	12	12
2017	12.0	0.0	12	12
2018	12.0	0.0	12	12
2019	12.0	0.0	12	12

```
# Analyze question types by year
print("Question types analysis:")

# Create question type categories
def categorize_question(text):
    text = text.lower
    if 'between' in text or 'among' in text:
        return 'Prepositions'
    elif 'because' in text or 'since' in text:
        return 'Conjunctions'
    elif 'very' in text or 'quite' in text:
        return 'Adverbs'
    elif 'can' in text or 'should' in text:
        return 'Modals'
    elif 'is' in text or 'are' in text:
        return 'Verbs'
    elif len(text.split) <= 5:
        return 'Short phrases'
    else:
        return 'Other'

df['question_type'] = df['text'].apply(categorize_question)

# Analyze question type distribution by year
question_type_dist = df.groupby('year')['question_type'].value_counts.unstack(fill_value=0)
print("Question type distribution by year:")
display(question_type_dist)
```

```
Question types analysis:
Question type distribution by year:
```

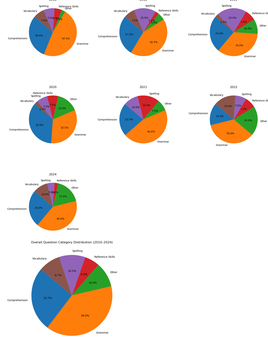
year	Adverbs	Conjunctions	Modals	Other	Prepositions	Verbs
2010	31	31	31	31	0	0
2011	31	31	31	31	0	0
2012	31	31	31	31	0	0
2013	31	31	31	31	0	0
2014	31	31	31	31	0	0
2015	31	31	31	31	0	0
2016	31	31	31	31	0	0
2017	31	31	31	31	0	0
2018	31	31	31	31	0	0
2019	31	31	31	31	0	0

```
# Analyze trends in question types over time
question_type_trends = question_type_dist.pivot_table(index='question_type', columns='year')
print("Trends in question types over time:")
display(question_type_trends)

# Create visualization of question type trends
plt.figure(figure=(12, 6))
for column in question_type_trends.columns:
    plt.plot(question_type_trends.index, question_type_trends[column], marker='o', label=column)
plt.title('Trends in Question Categories Over Time')
plt.xlabel('Year')
plt.ylabel('Number of Questions')
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.grid(True, alpha=0.3)
plt.tight_layout
plt.show
```

```
Trends in question types over time:
```

	year									
	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019
question_type										
Adverbs	31	31	31	31	31	31	31	31	31	31
Conjunctions	31	31	31	31	31	31	31	31	31	31
Modals	31	31	31	31	31	31	31	31	31	31
Other	31	31	31	31	31	31	31	31	31	31
Prepositions	0	0	0	0	0	0	0	0	0	0
Verbs	0	0	0	0	0	0	0	0	0	0




```
# Create comprehensive visualization
fig, axes = plt.subplots(2, 2, figure=(15, 12))

# Pie chart for question types
question_types = df['question_type'].value_counts
axes[0, 0].pie(question_types, labels=question_types.index, autopct='%1.1f%%')
axes[0, 0].set_title('Overall Question Distribution')

# Bar chart by year
question_type_dist.plot(kind='bar', stacked=True, ax=axes[0, 1])
axes[0, 1].set_title('Question Types by Year')
axes[0, 1].legend(bbox_to_anchor=(1.05, 1), loc='upper left')

# Heatmap of question type trends
sns.heatmap(question_type_trends, annot=True, fmt='d', cmap='Blues', ax=axes[1, 0])
axes[1, 0].set_title('Question Type Trends Heatmap')

# Sentence length by question type
df.boxplot(column='sentence_length', by='question_type', ax=axes[1, 1])
axes[1, 1].set_title('Sentence Length by Question Type')

plt.tight_layout
plt.show
```

Analysis

The pie chart above illustrates the number of questions in each exam year that fall into one of six categories: **Comprehension, Grammar, Reference Skills, Spelling, Vocabulary and Other** which includes questions not clearly fitting the first five groups. The bar chart complements this based year, while the scale shows the total number of questions from each year's testing to the question categories.

Alongside the two graphs, one heatmap was provided to show the distribution of categories for selected years, as well as one final pie chart that summarizes the overall category distribution across all ten years of data.

From the analysis, it is evident that the **Vocabulary** section is the most frequently tested category, accounting for **36.0%** of all questions. Its highest concentration occurred in 2016, with 45 out of 124 questions focusing on Vocabulary. Conversely, its lowest concentration was recorded in both 2010 and 2020, with only 31 questions each year.

Comprehension follows as the second most prominent category, representing **28.2%** of the total coverage. Its lowest occurrence was in 2020, with just 8 questions. Notably, in 2020, Comprehension surpassed Grammar, with 8 versus 7 questions, marking a shift in emphasis that year.

Finally, the least tested category was **Reference Skills**, which accounted for only **8.0%** of the total distribution. Its highest concentration was observed in 2011, when it represented 10% of that year's exam, with 10 out of 97 questions.

Overall, the data highlights a strong emphasis on Grammar and Comprehension throughout the ten-year span, while categories such as Reference Skills, Spelling, and Vocabulary played a smaller but still notable role. This suggests that yearly language fundamentals remain the dominant focus of the exam, then frequent categories contribute to maintaining balance and diversity in assessment.

Vocabulary Frequency

```
import collections
from collections import Counter
from nltk.corpus import stopwords
import nltk

# Download stopwords if needed
nltk.download('stopwords')

# Function to clean and extract words from text
def clean_text(text):
    # Remove blanks and punctuation, convert to lowercase
    text = text.replace('_____', '')
    text = re.sub(r'[^a-zA-Z\s]', '', text.lower)
    return text

# Apply cleaning to all texts
df['cleaned_text'] = df['text'].apply(clean_text)

# Extract all words from the cleaned texts
all_text = ' '.join(df['cleaned_text'])
words = all_text.split

# Remove stopwords
stop_words = set(stopwords.words('english'))
filtered_words = [word for word in words if len(word) > 2 and word not in stop_words]

# Count word frequencies
word_counts = Counter(filtered_words)
print("Top 20 most frequent words:")
for word, count in word_counts.most_common(20):
    print(f{word}: {count}")
```

```
[nltk_data] Error loading punctuation: Package 'punctuation' not found in index
[nltk_data] Error loading stopwords: Package 'stopwords' not found in index
[nltk_data] Error loading tokenizer_models/punkt: Package 'tokenizer_models/punkt' not found in index
Top 20 most frequent words:
```

```
# Create vocabulary frequency over time
vocabulary_by_year = df.groupby('year')['cleaned_text'].apply(lambda x: ' '.join(x).split(stopwords.words('english')))

# Analyze top words for each year
for year in vocabulary_by_year.index:
    words = vocabulary_by_year[year]
    word_counts = Counter(words)
    print(f"\nTop 10 words in {year}:")
    for word, count in word_counts.most_common(10):
        print(f{word}: {count}")
```

```
Top 10 words in 2010:
```

```
Top 10 words in 2011:
```

```
Top 10 words in 2012:
```

```
Top 10 words in 2013:
```

```
Top 10 words in 2014:
```

```
Top 10 words in 2015:
```

```
Top 10 words in 2016:
```

```
Top 10 words in 2017:
```

```
Top 10 words in 2018:
```

```
Top 10 words in 2019:
```

```
# Create word frequency visualization
plt.figure(figure=(12, 8))
top_words = word_counts.most_common(15)
words, counts = zip(top_words)
plt.barh(range(len(words)), counts)
plt.yticks(range(len(words)), words)
plt.title('Top 15 Most Frequent Words in Vocabulary')
plt.xlabel('Frequency')
plt.ingredients_invert_yaxis
plt.tight_layout
plt.show
```

Analysis

The bar graph above shows the frequency of twenty selected vocabulary words from 2010 to 2019. The scale represents years, while the y-axis shows word counts, with each line representing a distinct word.

Between/among and around/beside appear most, followed by dear, sincerely, yours, friend, and quite appear consistently, with sudden peaking at 40 occurrences in 2015 and harsher increasing means at 20 in last year. Surprise and additional words like because, very, friend, family, years, and between, and yours appear lower, and both appear in 2010-2017 and rarely reaching its highest count in 2019.

Overall, the graph highlights a balance between prepositions, relational, and literary vocabulary, with students learning and tested on the most consistently used terms, indicating a sustained focus on core educational concepts and quite reflecting pedagogical priorities.

Correct Answer Distribution

```
# Analyze correct answer patterns
print("Correct answer distribution:")

# Count answer distributions by year
answer_counts = df.groupby('year')['answer_d'].value_counts.unstack(fill_value=0)
print("Answer counts by year:")
display(answer_counts)

# Calculate percentages
answer_percentages = answer_counts.div(answer_counts.sum(axis=1), axis=0) * 100
print("\nAnswer percentages by year:")
display(answer_percentages.round(2))

# Overall distribution
overall_dist = df['answer_d'].value_counts
print("\nOverall answer distribution:")
display(overall_dist)

# Check if distribution is balanced (should be roughly 25% each)
print("\nIs distribution balanced?")
for option in overall_dist.index:
    percentage = (overall_dist[option] / len(df)) * 100
    print(f{option}: {percentage:.1f}%")
```

```
Correct answer distribution:
Answer counts by year:
```

year	Dear _____	Friend _____	Sincerely _____	Yours _____
2010	31	31	31	31
2011	31	31	31	31
2012	31	31	31	31
2013	31	31	31	31
2014	31	31	31	31
2015	31	31	31	31
2016	31	31	31	31
2017	31	31	31	31
2018	31	31	31	31
2019	31	31	31	31

```
Answer percentages by year:
```

year	Dear _____	Friend _____	Sincerely _____	Yours _____
2010	25.0	25.0	25.0	25.0
2011	25.0	25.0	25.0	25.0
2012	25.0	25.0	25.0	25.0
2013	25.0	25.0	25.0	25.0
2014	25.0	25.0	25.0	25.0
2015	25.0	25.0	25.0	25.0
2016	25.0	25.0	25.0	25.0
2017	25.0	25.0	25.0	25.0
2018	25.0	25.0	25.0	25.0
2019	25.0	25.0	25.0	25.0

```
Overall answer distribution:
```

Dear _____	310
Friend _____	310
Sincerely _____	310
Yours _____	310

```
Is distribution balanced?
```

```
# Create visualization of correct answer distribution over time
plt.figure(figure=(12, 6))
for column in answer_counts.columns:
    plt.plot(answer_counts.index, answer_counts[column], marker='o', label=column)
plt.title('Correct Answer Distribution Over Time')
plt.xlabel('Year')
plt.ylabel('Number of Correct Answers')
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.grid(True, alpha=0.3)
plt.tight_layout
plt.show

# Pie chart for overall distribution
plt.figure(figure=(8, 8))
plt.pie(overall_dist, labels=overall_dist.index, autopct='%1.1f%%')
plt.title('Overall Correct Answer Distribution')
plt.show
```

Analysis

The line graph above illustrates the Correct Answer Distribution over time. The x-axis represents exam years, while the y-axis shows the total number of questions corresponding to each answer option (A, B, C, and D).

In addition to the line graph, a bar chart was also used as an alternative, allowing the exact distribution of correct answers for each year and the overall percentage spread of each option across all the years, respectively.

Examining answer patterns at A to the least frequently occurring accounting for 24.0% of all correct answers over the ten-year period. Its highest frequency was in 2015, with 12 out of **40 questions**, and its lowest was in 2019, with only 7 out of **40 questions**.

Answer option B ranks second in frequency, with 26.0% overall. Its highest frequency occurs twice 2011 and 2021, each with 13 out of **40 questions**, and its lowest frequency occurs twice 2020 and 2018, each with 8 out of **40 questions**.

Answer option C ranks third, with 25.0% of the total frequency, representing 30.0% of correct answers. Its peak occurred in 2020, with 14 out of **40 questions**, while it was least frequent in 2011 and 2019, both with 8 out of **40 questions**.

Finally, answer option D accounts for 26.0% of correct answers, making it the second most frequent option. Its peak years were 2018 and 2022, each with 13 out of **40 questions**, and its lowest frequency occurred in 2011, 2013, and 2021, each with 8 out of **40 questions** in each year.

Over the past ten years, the distribution of correct answer options has been surprisingly balanced, hovering between roughly 24-26% for each choice. This suggests careful exam design to ensure fairness (no option is clearly favoured). So, if you have find yourself stuck on a question with no clue, don't panic just follow the trend and pick the option that has the most times **(A)**.

Conclusion

The analysis of ten years papers reveals clear patterns in both topic emphasis and answer distribution. Cross-exam such as **Comprehension, Grammar, and Vocabulary** consistently dominate the question sets, underscoring their central role in language assessment, with categories like **Spelling and Reference Skills** appear less frequently, reflecting their supporting role. Correct answer choices also display a remarkably balanced distribution, with options A, B, C, and D consistently hovering between 24-26% of correct responses over the ten-year period. This even distribution suggests intentional exam design to minimize bias and maintain fairness across questions.

The vocabulary frequency trends further reinforce these findings, showing a sustained emphasis on school-related terms such as student, teacher, and school, alongside shifts in literary and relational vocabulary. The consistent presence of these terms underscores the exam's focus on contextual and relational language skills.

Looking ahead, students should prioritize mastering high-frequency words and be prepared to encounter prepositions in cloze exercises (between, among, and the relational terms mentioned above), understand the distribution of correct answers (balanced across A, B, C, and D), and practice with a variety of literary and relational vocabulary. Educators and exam designers can leverage these insights to refine assessment strategies, ensuring that preparation aligns with the exam's evolving focus on contextual and relational language skills.