MIDTERM PROJECT

PROJECT: NATURAL LANGUAGE PROCESSING

GROUP :

CAO NGUYEN THAI THUAN – 522H0092

NGUYEN PHAM QUANG HUY – 522H0138

Instructor: Associate Professor, Ph.D. LE ANH CUONG

Outline

- Categorize questions about players, leagues and teams in football
- Build synthetic datasets using Large Language Models
- Common methods of text representation against traditional ML methods
- Doc2Vec in action – comparison between pretrained and home-trained

Categorize questions about players, leagues and teams in football

In the field of football, automated question-answering systems and sports chatbots are increasingly being widely applied. One of the critical challenges in this domain is classifying questions into categories such as players, tournaments, and teams. Automatically categorizing questions enhances information retrieval efficiency, improves user experience, and optimizes virtual assistant systems.

Categorize questions about players, leagues and teams in football

The main objectives of this study include:

- 1.Building a dataset of questions related to players, tournaments, and football teams.
- 2.Applying machine learning methods to classify questions into predefined categories.
- 3.Evaluating the performance of each method and proposing an optimal solution.

Categorize questions about players, leagues and teams in football

Benefits of Datasets:

- Building a suitable dataset for classifying football-related questions offers several key benefits:
- Optimizing search systems: Accurately categorized questions allow systems to quickly provide relevant answers, enabling users to find information more easily.
- Improving sports chatbots: Chatbots can better understand question contexts, deliver more precise responses, and enhance user experience.

Categorize questions about players, leagues and teams in football

Benefits of Datasets:

- Supporting AI model training: The dataset helps train machine learning models to accurately identify question topics, which can then be applied to other applications like virtual assistants or sports analytics.
- Developing sports data analytics applications: AI systems can leverage this data to track search trends, predict popular topics, and generate insights within the football community.

Build synthetic datasets using Large Language Models

Step 1: Define the Dataset Objective

- This dataset is designed to train an NLP model to process football-related questions

- The dataset must be diverse and include various types of questions.

Build synthetic datasets using Large Language Models

Step 2: Identify Main Categories

- We divide the dataset into key categories:

- Football history

- Players and coaches

- Tournaments and clubs

- Football rules

- Statistics and records

Build synthetic datasets using Large Language Models

Step 3: Build a Keyword List

Create a list of important keywords for each category. Example for "Football History": keywords may include "World Cup," "Euro," "legend," "final," etc.

```
question_keywords = {
    "Thông tin chung": ["bóng đá là gì", "luật việt vị", "VAR", "thời gian thi đấu", "trọng tài", "hệ thống giải đấu", "fair-play
    "Chiến thuật": ["phòng ngự phản công", "pressing", "tấn công trung lộ", "sơ đồ 4-3-3", "sơ đồ 5-3-2", "chiến thuật tiki-taka"
    "Cầu thủ & Đội bóng": ["Messi", "Ronaldo", "Erling Haaland", "Mbappe", "Real Madrid", "Barcelona", "Manchester United", "câu
    "Lịch sử": ["World Cup đầu tiên", "Euro đầu tiên", "Champions League", "trận đấu nhiều bàn thắng nhất", "cầu thủ ghi nhiều bà
    "Sự kiện & Giải đấu": ["Champions League", "Euro", "World Cup 2022", "Copa America", "Asian Cup", "V-League", "Premier League
    "Phân tích trận đấu": ["sơ đồ chiến thuật", "xG (expected goals)", "đánh giá cầu thủ", "thống kê chuyền bóng", "số lần dứt đi
    "Huấn luyện viên": ["Pep Guardiola", "Mourinho", "Sir Alex Ferguson", "Carlo Ancelotti", "chiến thuật huấn luyện viên", "vai
    "Kỹ năng cá nhân": ["sút bóng", "chuyền bóng", "kiểm soát bóng", "đánh đầu", "cách rê bóng", "cách sút phạt", "cách cản phá",
    "Chiến thuật đội bóng": ["tiki-taka", "gegenpressing", "catenaccio", "đội hình pressing", "tấn công tổng lực", "cách xây dựng
    "Phong cách thi đấu": ["bóng đá đẹp", "bóng đá thực dụng", "bóng đá phòng ngự", "tấn công tổng lực", "bóng đá phản công nhanh
    "Chuyển nhượng & Thị trường": ["thị trường chuyển nhượng", "các vụ chuyển nhượng đắt giá nhất", "luật chuyển nhượng FIFA", "c
    "Cổ động viên & Văn hóa bóng đá": ["Ultras", "hooligan", "bài hát cổ động", "sự cuồng nhiệt của fan", "cách cổ vũ đội bóng",
    "Sự nghiệp cầu thủ": ["con đường trở thành cầu thủ chuyên nghiệp", "đào tạo cầu thủ trẻ", "giải nghệ", "chấn thương trong bón
    "Công nghệ trong bóng đá": ["VAR", "goal-line technology", "phân tích dữ liệu cầu thủ", "thiết bị đo lường thể chất", "công n
    "Ảnh hưởng xã hội của bóng đá": ["bóng đá và chính trị", "tác động kinh tế của bóng đá", "các tổ chức bóng đá từ thiện", "bón
}
```

Build synthetic datasets using Large Language Models

Step 4: Construct Question Templates

Develop question templates for each category.

```
templates = {
      "Thông tin chung": [
        f"Bạn có thể giải thích {keyword} không?",
        f"{keyword} có vai trò gì trong bóng đá?",
        f"Luật {keyword} áp dụng như thế nào?",
        f"{keyword} ảnh hưởng thế nào đến trận đấu?",
        f"Làm thế nào để hiểu rõ hơn về {keyword}?",
        f"Bạn có thể cung cấp một cái nhìn toàn diện về {keyword}, bao gồm nguồn gốc, sự phát triển và cách nó được áp dụng t
        f"Trong lịch sử bóng đá, {keyword} đã từng gây ra những tranh cãi hoặc thay đổi lớn nào? Có ví dụ nào về những trận c
        f"{keyword} có phải là một phần quan trọng trong sự phát triển của bóng đá không? Nó có liên quan gì đến những thay c
        f"Những hiểu lầm phổ biến về {keyword} là gì? Tại sao nhiều người có quan điểm trái chiều về {keyword}, và đâu là nhữ
    ],
```

Build synthetic datasets using Large Language Models

Step 5: Generate Data Automatically

Use Python programming to combine keywords and templates into various questions.

```
158     question = np.random.choice(templates.get(category, [f"Thông tin về {keyword}?"]))
159     question_corpus.append((question, category))
160
161 # Save the generated dataset
162 output_df = pd.DataFrame(question_corpus, columns=["text", "label"])
163 output_file = "/content/sample_data/generated_soccer_questions.csv"
164 output_df.to_csv(output_file, index=False)
165
166 print(f"Dataset generated and saved to {output_file}")
167 print(f"Number of samples: {len(output_df)}")
```

Build synthetic datasets using Large Language Models

Step 6: Review and Refine the Data

In this step, we need to achieve:

- Remove duplicate or illogical questions.
- Check grammar and question accuracy.
- Add more keywords to enhance diversity.

Build synthetic datasets using Large Language Models

Step 7: Data Preprocessing

Convert text to lowercase.

Remove punctuation and numbers.

Tokenize words using NLP techniques.

```
def clean_text(self):
    self.df["clean_text"] = self.df["text"].str.lower()
    self.df["clean_text"] = self.df["clean_text"].str.replace(r"[{}]".format(string.punctuation), "", regex=True)
    self.df["clean_text"] = self.df["clean_text"].str.replace(r"\d+", "", regex=True)
    self.df["clean_text"] = self.df["clean_text"].apply(lambda x: " ".join(word_tokenize(str(x))))  # ✅ Fix: Use only undert
```

Build synthetic datasets using Large Language Models

2. Data Splitting

Split the dataset into training (80%) and testing (20%) sets.

```
28 # Sample 80% of data
29 df = df.sample(frac=0.8, random_state=42)
30 df.reset_index(drop=True, inplace=True)
31
32 print("Dữ liệu mẫu:")
33 print(df.head())
34
```

Build synthetic datasets using Large Language Models

3. Text Vectorization and Label Encoding

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

# One-Hot Encoding
X_one_hot_train, X_one_hot_test = processor_train.one_hot_encoding(y_train, y_test)

# Shape Summary
print("TF-IDF Shape:", X_tfidf_train.shape)
print("BoW Shape:", X_bow_train.shape)
print("One-Hot Shape:", X_one_hot_train.shape)

```

Common methods of text representation

Three models are trained and evaluated:

- Logistic Regression
- Naïve Bayes
- Decision Tree

Common methods of text representation

Step 1: Encoding the Data

Text data is transformed using BoW, TF-IDF, and One-Hot Encoding.

Labels are encoded using LabelEncoder to fit the data into the models properly.

Common methods of text representation

Step 2: Training and Evaluating Models

- Each model is trained on all three text representations.
- Accuracy (accuracy_score) and classification reports (classification_report) are recorded.

Common methods of text representation

Step 3: Comparing Results

- Accuracy scores for each model and representation method are stored.
- A bar chart visualizes model performance.

Common methods of text representation

BoW, TF-IDF and One – hot Encoding

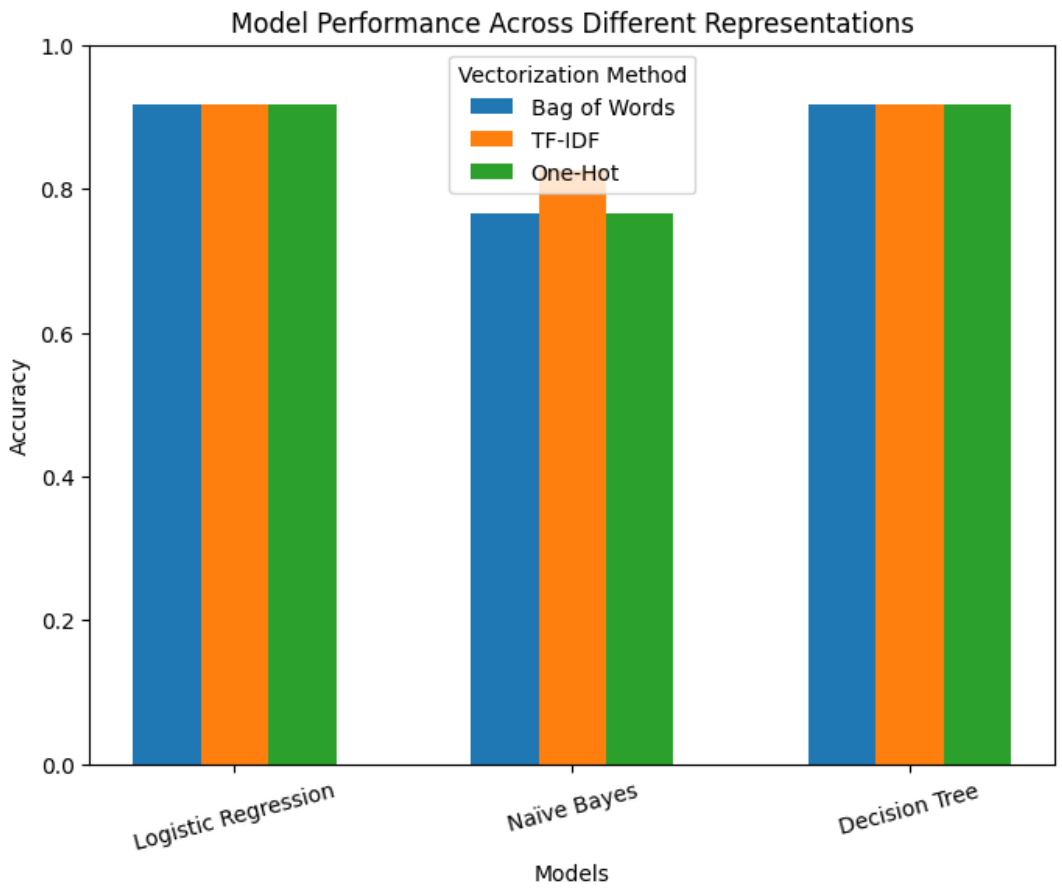
- BoW: Maps all unique occurrences of each word from a document.
- TF – IDF: Product of the rate of appearance of each term and the logarithmic value of the inverse document's term rate in the corpus.
- One-hot Encoding: a binary vectorization technique where for each present word in the vocab is noted 1 and vice versa.

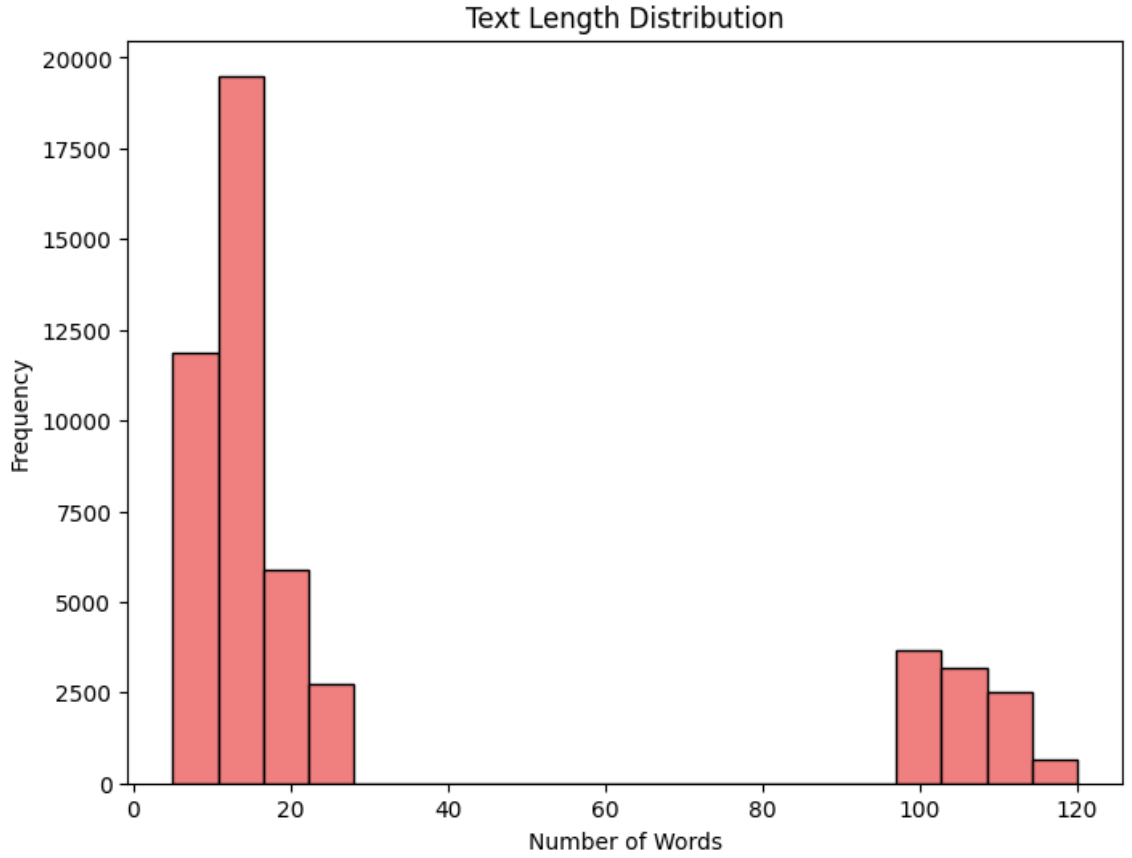
LabelEncoder:

- A vectorization technique where each unique word is a unique integer.

Common methods of text representation

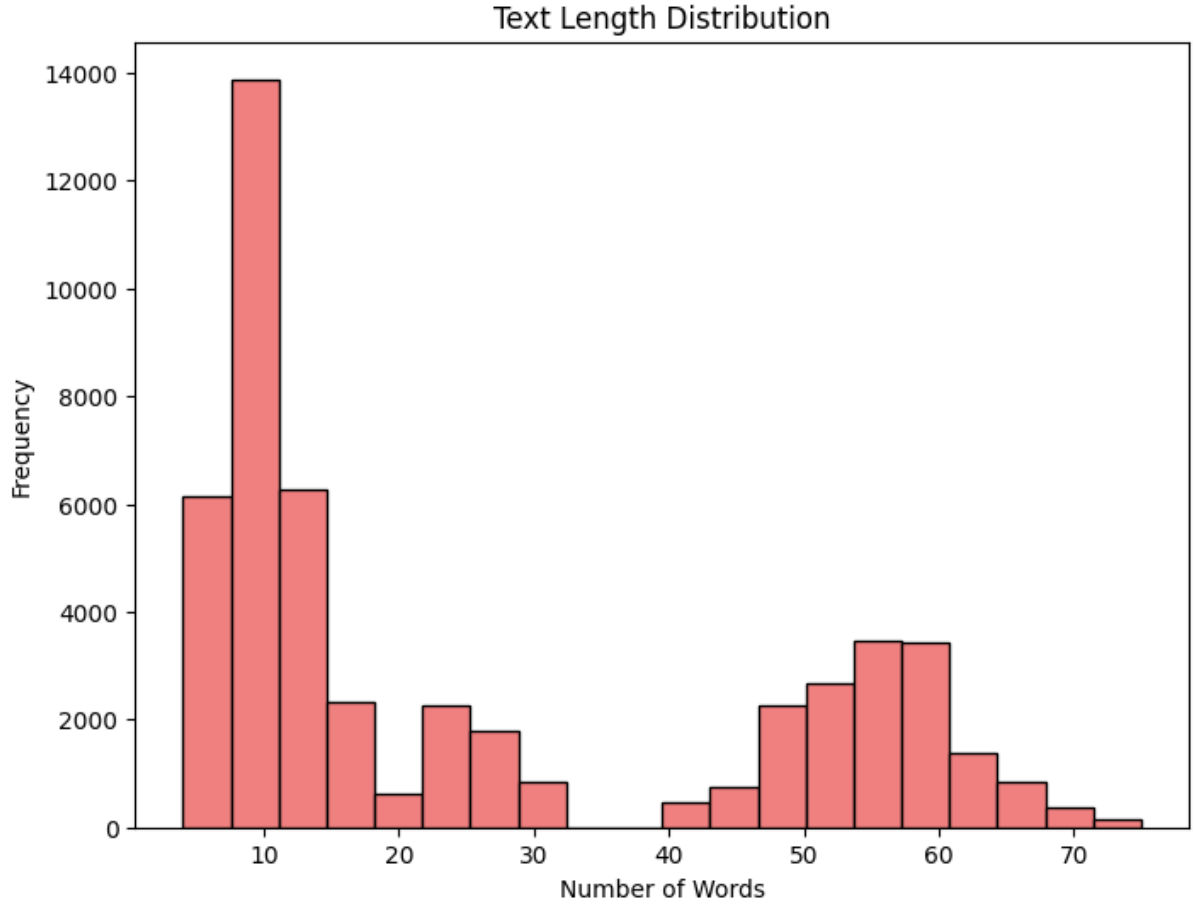
Comparison of accuracies between models(high short – high long):





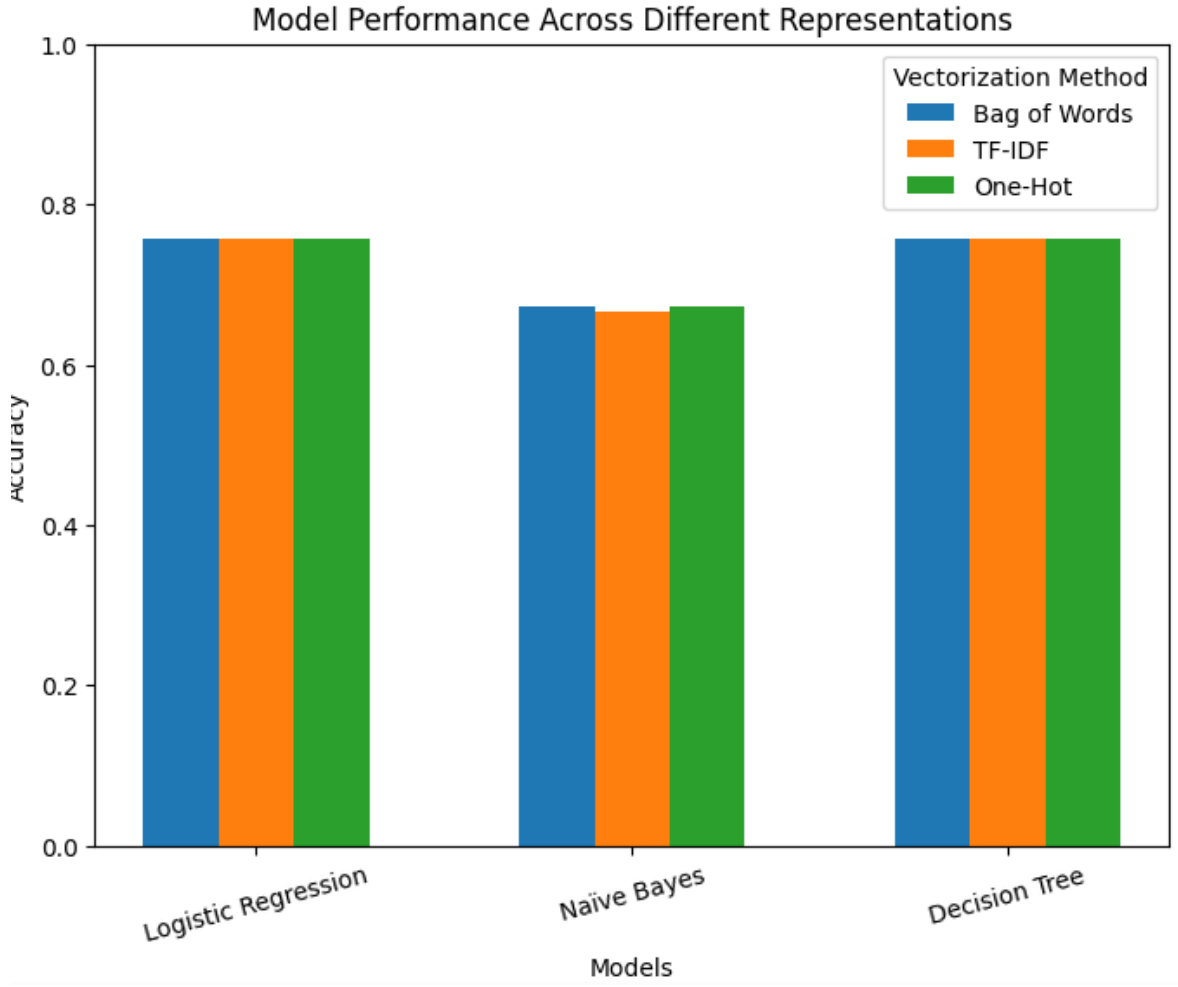
Common methods of text representation

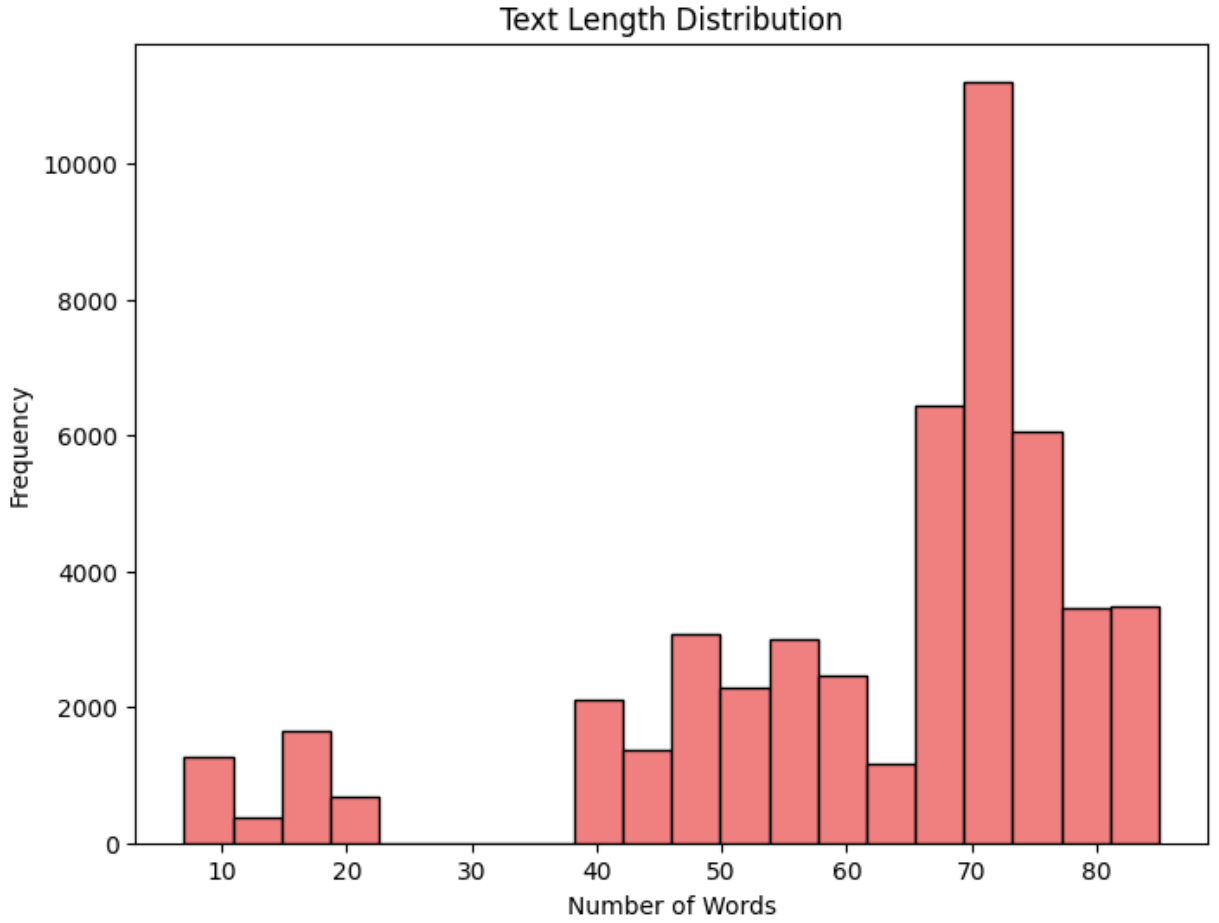
Comparison of accuracies between models(high short – high long – alt dataset):



Common methods of text representation

Comparison of accuracies between models(better balanced – alt dataset):





Common methods of text representation

Conclusion:

- These models and vectorization techniques have a harder time at handling longer sentences

- Between topics, templates variability and template length, the template length has the largest impact

- TF – IDF, while being the best vectorization method, also struggles with longer sentences

Doc2Vec

Why not BoW, TF - IDF or One – hot encoding?

- Doesn’t carry order or context

- Calculation resources scales with the size of the corpora

- Struggles with out-of-vocab words

Doc2Vec

The aim of Doc2Vec:

- Reduce high-dimensionality by creating dense, low dimensional vectors that captures meaning and context

- Can handle variable length and new or strange words

- Easier to fine tune

Doc2Vec

Doc2Vec – pretrained vs house-trained

```python
# Define PyTorch Doc2Vec Model
class SimpleDoc2Vec(nn.Module):
    def __init__(self, vocab_size, embedding_dim):
        super(SimpleDoc2Vec, self).__init__()
        self.embedding = nn.Embedding(vocab_size, embedding_dim, padding_idx=0)
        self.fc = nn.Linear(embedding_dim, 50)
        self.bn = nn.BatchNorm1d(50)  # Normalize embeddings for better stability

    def forward(self, x):
        embedded = self.embedding(x)  # Shape: (batch, seq_len, embedding_dim)
        x = embedded.mean(dim=1)  # Average over the sequence length
        x = self.bn(self.fc(x))  # Apply batch normalization
        return x
```

```
Gensim Doc2Vec Accuracy: 0.5944
PyTorch Doc2Vec Accuracy: 0.0550
```

Doc2Vec

Conclusion:

- Very hard to train yourself

- Doc2Vec is a complex but more versatile option for a wider range of problems

- Very dependent on a large corpora for better accuracy

Reference Document

Breiman, L. (2001). Random Forests. Machine Learning, 45(1), 5-32.

Quinlan, J. R. (1986). Induction of Decision Trees. Machine Learning, 1(1), 81-106.

Guyon, I., & Elisseeff, A. (2003). An Introduction to Variable and Feature Selection. Journal of Machine Learning Research, 3, 1157-1182.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springe

- Subject: Machine Learning

Thanks for your watching

This is the end of my group’s report