AG NEWS Category Classification Using NLP

Milestone 4: Project – Implement Embedding Methods

Student: Sudhish Subramaniam

+12368634776

subramaniam.su@northeastern.edu

Percentage of Effort Contributed by Student : 100%

Signature of Student : [illegible]

Submission Date:____5th November, 2023________

Introduction

In the rapidly evolving landscape of digital news consumption, the efficient categorization of news articles has become a paramount challenge for news platforms. The objective at hand is the development of a robust Natural Language Processing (NLP) model, a technological solution aimed at automating the categorization of news articles into distinct topics, namely "World," "Sports," "Business," and "Sci/Tech." The significance of this task lies in its potential to revolutionize the news industry by enhancing user experience and streamlining content recommendation processes. In an era where information is abundant and attention spans are limited, ensuring that readers can effortlessly access news content that aligns with their interests is a pressing concern. My mission is to create a sophisticated NLP model capable of classifying news articles with precision, relying on the analysis of their content and context. This classification is not merely an organizational endeavor but a pivotal means of optimizing content delivery and personalizing news recommendations. By automating this intricate process, I aim to empower news platforms to engage readers more effectively, potentially leading to increased user retention and overall satisfaction. In doing so, I embark on a journey to redefine how news is presented and consumed in the digital age.

Feature extraction and embedding methods techniques are fundamental in enabling NLP models to understand and process the complex nuances of textual data, such as news articles. Feature extraction involves transforming raw text into numerical representations that can be fed into neural networks. This process is crucial for capturing essential patterns and structures within the text, allowing the model to make informed decisions about article categorization. On the other hand, embedding methods, like Word2Vec, Fasttext , Glove, provide a way to represent words or phrases as dense, continuous vectors, preserving semantic relationships. This not only aids in understanding context but also helps the model generalize better across different texts. In the context of our mission to categorize news articles effectively, these techniques play a pivotal role in ensuring that our NLP model can analyze the content and context of news articles with precision and reliability, ultimately contributing to the improved user experience and content personalization.

Problem Statement

In the ever-evolving landscape of digital news consumption, news platforms face a critical challenge: efficiently categorizing news articles into distinct topics, such as "World," "Sports," "Business," and "Sci/Tech." The goal of this project is to develop a robust Natural Language Processing (NLP) model capable of automating this classification and categorization process. The significance of this endeavor lies in its potential to revolutionize the news industry by improving user experience and streamlining content recommendation processes. In an era characterized by information abundance and limited attention spans, providing readers with easy access to news content aligned with their interests is of utmost importance. The mission is to create an advanced NLP model that can classify news articles with precision by analyzing their content and context with nuance. This classification is not merely organizational but also pivotal for optimizing content delivery and personalizing news recommendations. By automating this complex process, the aim is to empower news platforms to engage readers more effectively, potentially leading to increased user retention and overall satisfaction, thus redefining how news is presented and consumed in the digital age.

Objectives of the project

- Create a NLP model that leverages deep learning techniques to accurately classify news articles into predefined topics.
- Achieve a high level of precision and recall in categorizing news articles, ensuring that the model can reliably assign articles to the correct topics.
- Implement advanced feature extraction and embedding methods to enable the model to capture the nuances and context of news articles effectively.
- Design the model for scalability, ensuring it can handle a large volume of news articles efficiently, with minimal computational resources.
- Categorizing news in a such a way that businesses can use it as a content recommendation system that utilizes the categorization model to provide readers with news content tailored to their interests, enhancing the overall user experience.

Data Used

Dataset Overview

The dataset at the core of this project is the AG News Corpus, a meticulously curated collection of news articles derived from an extensive pool of over 1 million news articles sourced from more than 2000 news outlets. This vast and diverse collection of articles has been accumulated over the course of a year through the efforts of the ComeToMyHead academic news search engine, which has been in operation since July 2004.

Specifically, the dataset used for this project is the AG's news topic classification dataset, curated by Xiang Zhang. This dataset has gained prominence as a benchmark for text classification tasks and serves as a well-established resource for addressing the news categorization problem. It comprises 30,000 training samples and 1,900 test samples per class. The dataset's substantial size and the balanced distribution of samples across categories make it an invaluable resource for training, fine-tuning, and evaluating Natural Language Processing (NLP) models, which are central to solving the news article categorization challenge.

Data Dependency

- **Class Balance Dependency:** The dataset's balanced distribution across "World," "Sports," "Business," and "Sci/Tech" categories is essential to prevent bias.

- **Textual Dependency:** The model relies on keywords and patterns in the news articles for categorization. Text content strongly influences predictions.

- **Train Test Quality Dependency:** Both training and test sets must be of high quality and representative to achieve reliable model performance.

- **Temporal Influence:** Temporal trends or seasonality in news topics may impact categorization accuracy.

- **Feature Influence:** The choice and quality of features (e.g., word embeddings) depend on the dataset's content.

- **Evaluation Metric Choice:** The dataset's class distribution affects the selection of appropriate evaluation metrics.

Managing these dependencies will ensure robust and accurate news article categorization.

Analysis

Data Preprocessing and transformation

The objective of this project is to automate the classification of news articles into predefined categories, namely "World," "Sports," "Business," and "Sci/Tech." This automated classification is crucial for improving content organization and personalizing news delivery on digital platforms, aiming to enhance user engagement and satisfaction.

Data preprocessing plays a vital role in any natural language processing (NLP) task. In this project, I carried out several preprocessing steps to prepare the news article data for classification. These steps included converting all text to lowercase, removing punctuation and numbers, eliminating common English stopwords, applying lemmatization to reduce word dimensionality, tokenizing the text, and padding sequences for uniform length.

Following data preprocessing, I built a machine learning model for news article classification. The model utilized a vocabulary size of 10,000 and an embedding size of 32 to create word-level representations of the text data. This embedding layer helps capture semantic relationships between words in news articles.

In conclusion, I have successfully preprocessed and transformed news article data, making it ready for classification. Our machine learning model, which uses word embeddings and sequence padding, is prepared to classify news articles into "World," "Sports," "Business," and "Sci/Tech" categories. This classification will improve content organization and personalization on digital platforms, ultimately enhancing user engagement and satisfaction.

Explanatory data analysis

The dataset for this analysis consists of 120,000 samples in the training set and 7,600 samples in the test set, each comprising two columns: 'text' and 'label.' The 'label' column represents the category of news, with each of the four classes (0, 1, 2, and 3) containing 30,000 samples. This can be seen in Fig. 1. This balanced class distribution ensures a fair representation of different news categories. Moreover, there are no missing values in either the training or test dataset, indicating data integrity.

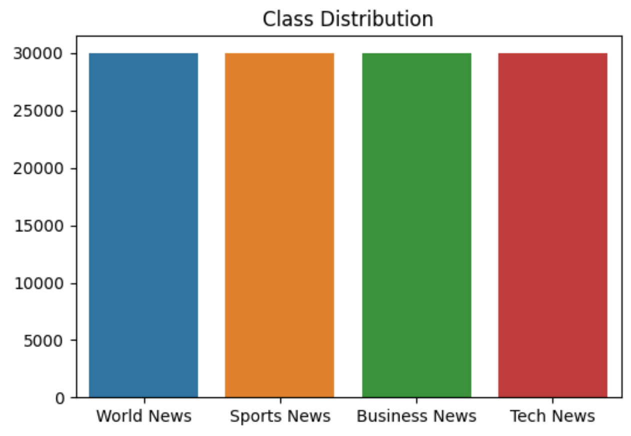


Fig. 1.

Word cloud visualizations were created for each news category, revealing the most common terms associated with each. In the 'World News' category, terms such as "prime minister," "iraq," and "israel" were prevalent. 'Sports News' featured terms like "game," "season," and "team." In the 'Business News' category, words like "company," "price," "oil," and "stocks" dominated. 'Science and Technology News' highlighted terms such as "microsoft," "google," "email," and "internet." These insights provide a preliminary understanding of the vocabulary within each category. These prevalent words are clearly seen in Fig. 2 (a), (b), (c), (d).

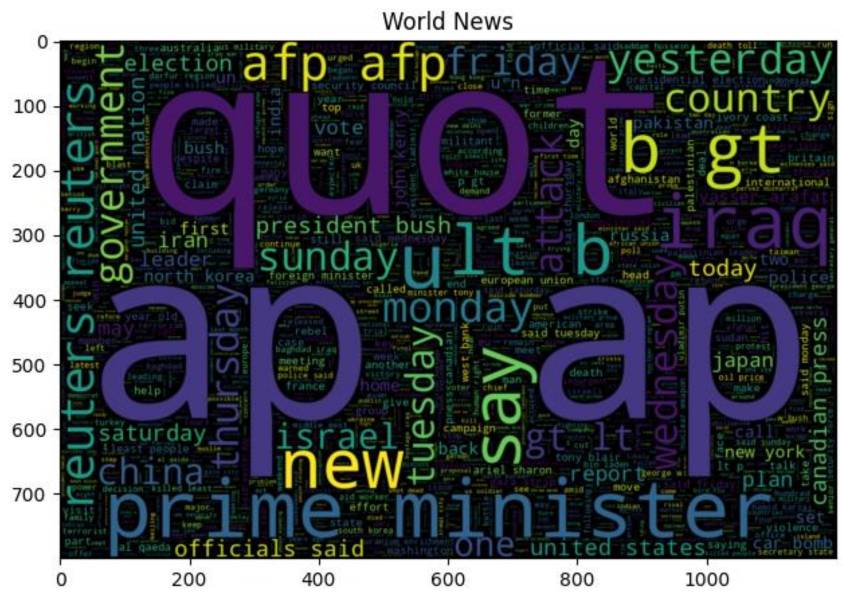


Fig 2 (a). Word Cloud of World News

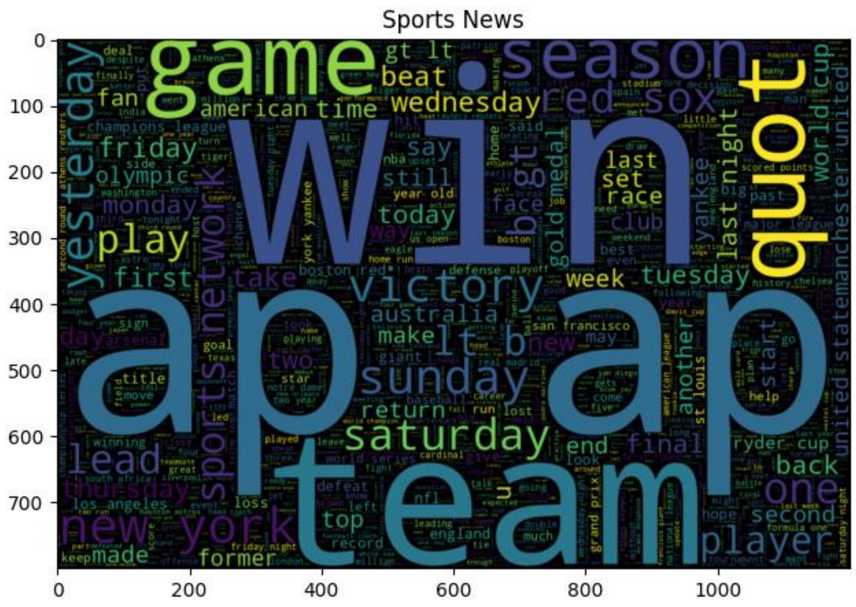


Fig 2 (b). Word Cloud of Sports News

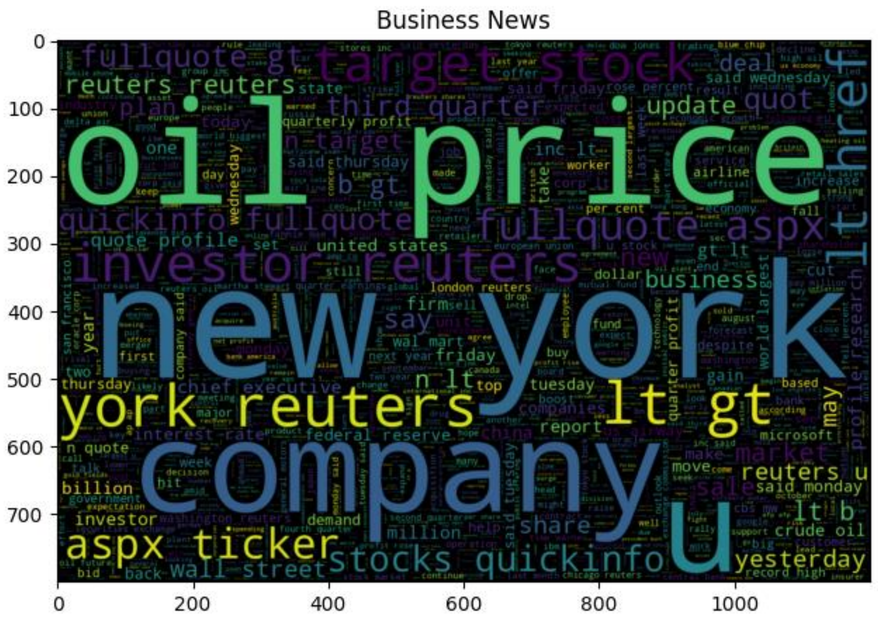


Fig 2 (b). Word Cloud of Business News

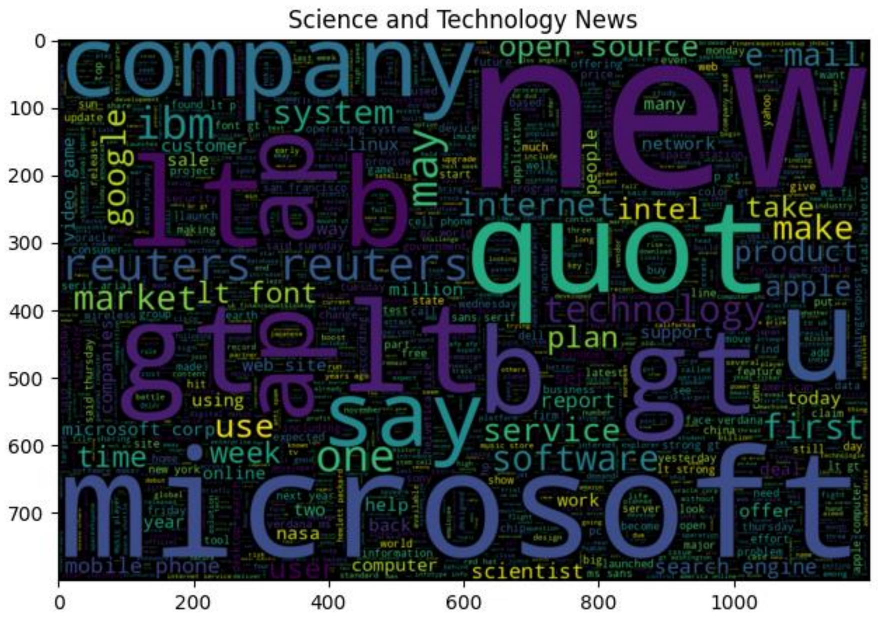
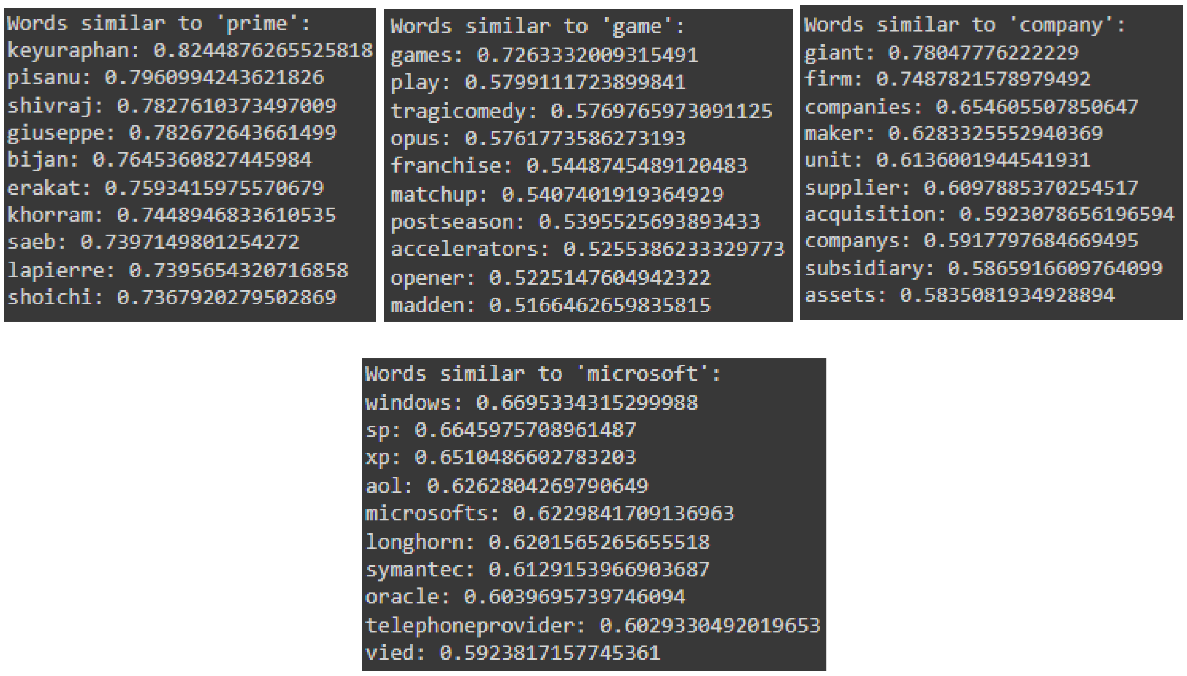


Fig 2 (d). Word Cloud of Science and Technology News

A Word2Vec model was trained on the text data to capture word embeddings. The model used a vector size of 100, a window of 5, and a minimum word count of 1. This model allows for the exploration of semantic relationships between words. For example, words like "prime" included "shivraj," "keyuraphan," and others. Similarly, words related to "game" were "games," "play," and others. For "company," terms like "giant" and "firm" were identified. For “Microsoft”, terms like “windows”, “microsofts”, “oracle” were identified. The model can be a valuable resource for understanding word associations within the dataset. This can be seen in Fig. 3.



```
Words similar to 'prime':
keyuraphan: 0.8244876265525818
pisanu: 0.7960994243621826
shivraj: 0.7827610373497009
giuseppe: 0.782672643661499
bijan: 0.7645360827445984
erakat: 0.7593415975570679
khorram: 0.7448946833610535
saeb: 0.7397149801254272
lapierre: 0.7395654320716858
shoichi: 0.7367920279502869
```

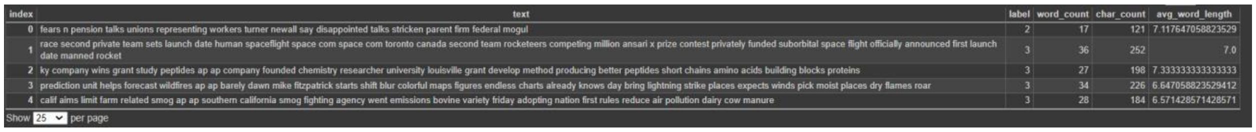
```
Words similar to 'game':
games: 0.7263332009315491
play: 0.5799111723899841
tragicomedy: 0.5769765973091125
opus: 0.5761773586273193
franchise: 0.5448745489120483
matchup: 0.5407401919364929
postseason: 0.5395525693893433
accelerators: 0.5255386233329773
opener: 0.5225147604942322
madden: 0.5166462659835815
```

```
Words similar to 'company':
giant: 0.78047776222229
firm: 0.7487821578979492
companies: 0.654605507850647
maker: 0.6283325552940369
unit: 0.6136001944541931
supplier: 0.6097885370254517
acquisition: 0.5923078656196594
companys: 0.5917797684669495
subsidiary: 0.5865916609764099
assets: 0.5835081934928894
```

```
Words similar to 'microsoft':
windows: 0.6695334315299988
sp: 0.6645975708961487
xp: 0.6510486602783203
aol: 0.6262804269790649
microsofts: 0.6229841709136963
longhorn: 0.6201565265655518
symantec: 0.6129153966903687
oracle: 0.6039695739746094
telephoneprovider: 0.6029330492019653
vied: 0.5923817157745361
```

Fig. 3. Similar Words using Word2Vec

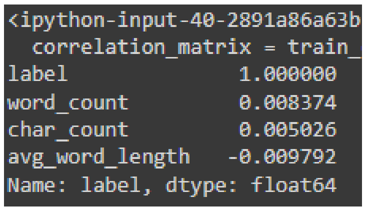
Text length analysis was conducted, calculating word count, character count, and average word length for each sample in the training and test datasets (shown in Fig. 4). Correlation analysis (Show in Fig. 5) between these text length features, and the 'label' column indicated very weak correlations, suggesting that text length may not be a strong predictor of news category. This observation highlights the importance of other features in classifying news.



index	text	label	word_count	char_count	avg_word_length
0	fears n pension talks unions representing workers turner newall say disappointed talks stricken parent firm federal mogul	2	17	121	7.117647058823529
1	race second private team sets launch date human spaceflight space com space com toronto canada second team rocketeers competing million ansari x prize contest privately funded suborbital space flight officially announced first launch date manned rocket	3	36	252	7.0
2	ky company wins grant study peptides ap ap company founded chemistry researcher university louisville grant develop method producing better peptides short chains amino acids building blocks proteins	3	27	198	7.333333333333333
3	prediction unit helps forecast wildfires ap ap barely dawn mike fitzpatrick starts shift blur colorful maps figures endless charts already knows day bring lightning strike places expects winds pick moist places dry flames roar	3	34	226	6.647058823529412
4	calif aims limit farm related smog ap ap southern california smog fighting agency went emissions bovine variety friday adopting nation first rules reduce air pollution dairy cow manure	3	28	184	6.571428571428571

Show 25 per page

Fig. 4. Head rows with word count, character count and average word length for each news



```
<ipython-input-40-2891a86a63b
  correlation_matrix = train_
label              1.000000
word_count         0.008374
char_count         0.005026
avg_word_length   -0.009792
Name: label, dtype: float64
```

Fig. 5. Correlation between news and other columns

In the NER Analysis, I have observed the following:

1. The dataset seems to contain information about various organizations, including technology companies like Microsoft, Google, IBM, and others such as Reuters, NASA, and the United Nations. This indicates that the news articles might be related to developments in these organizations.

2. The presence of individuals like George W. Bush, Vladimir Putin, and Michael Phelps suggests that the news articles may cover political figures, celebrities, or prominent individuals in various fields.

3. Locations and countries such as Iraq, New York, and China are recognized. This suggests that the news dataset may include articles on international affairs and global events.

4. The identification of dates and times like "Tuesday," "August," and "last week" implies that the news articles might be categorized based on the time of occurrence or publication.

Based on this NER analysis, I can infer that the news dataset likely covers a wide range of topics, including technology, politics, international affairs, and events that occurred at specific times. Categorizing the news articles into these four categories could provide the model a structured approach for analyzing and organizing the dataset.

In summary, the analysis provides insights into the dataset's composition, word cloud visualizations for different news categories, and the training of a Word2Vec model to understand word similarities. The examination of text length features suggests that these factors may have limited predictive power for news category classification.

Feature engineering and feature selection

Feature engineering is the process of transforming raw text data into numerical features, enabling the training of machine learning models. The initial step involves using the TfidfVectorizer from scikit-learn to convert the text data into TF-IDF vectors, which serve as essential features. However, one noteworthy consideration is the limitation on the number of features due to system constraints. The code imposes a cap of 4500 features to avoid system crashes, a common occurrence when dealing with high-dimensional data.

Following the TF-IDF vectorization, the code goes on to extract the vocabulary and calculates the word frequencies in the training dataset. The term frequencies are particularly informative as they reveal which words or terms hold the most significance within the dataset. Sorting the features by frequency in descending order, the code presents the top 50 features, shedding light on the most influential terms. This step is pivotal for feature selection and model interpretability.

Some most important features include: “new”, “said”, “world”, “company”, “Microsoft”, “iraq”, “oil”. These features are relevant to the world news and can be considered as most important features during modelling. This shows that the text is rightly processed.

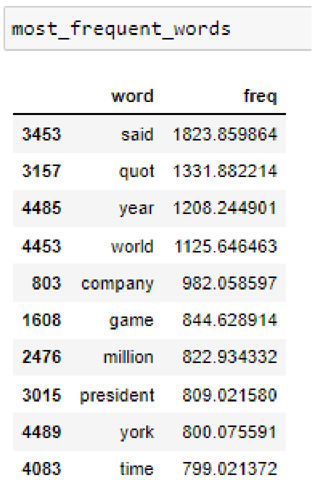
To make the features ready for machine learning, the TF-IDF vectors are transformed into arrays and then converted into Pandas DataFrames. These DataFrames house the features for both the training and testing datasets. In conclusion, this code snippet emphasizes the importance of feature engineering in preparing text data for machine learning. By setting a limit of 4500 features, it effectively addresses system constraints, ensuring the stability of the computational environment.

Implementing Embedding Layer

This report explores the use of word embedding models, specifically Word2Vec, GloVe, and FastText, to find similar words and assess their cosine similarity to identify the best embedding model among them. Word embedding is a crucial technique in Natural Language Processing (NLP) that helps represent words as vectors in a continuous space, capturing semantic relationships between words. Cosine similarity is used to measure the similarity between word vectors.

Word2Vec

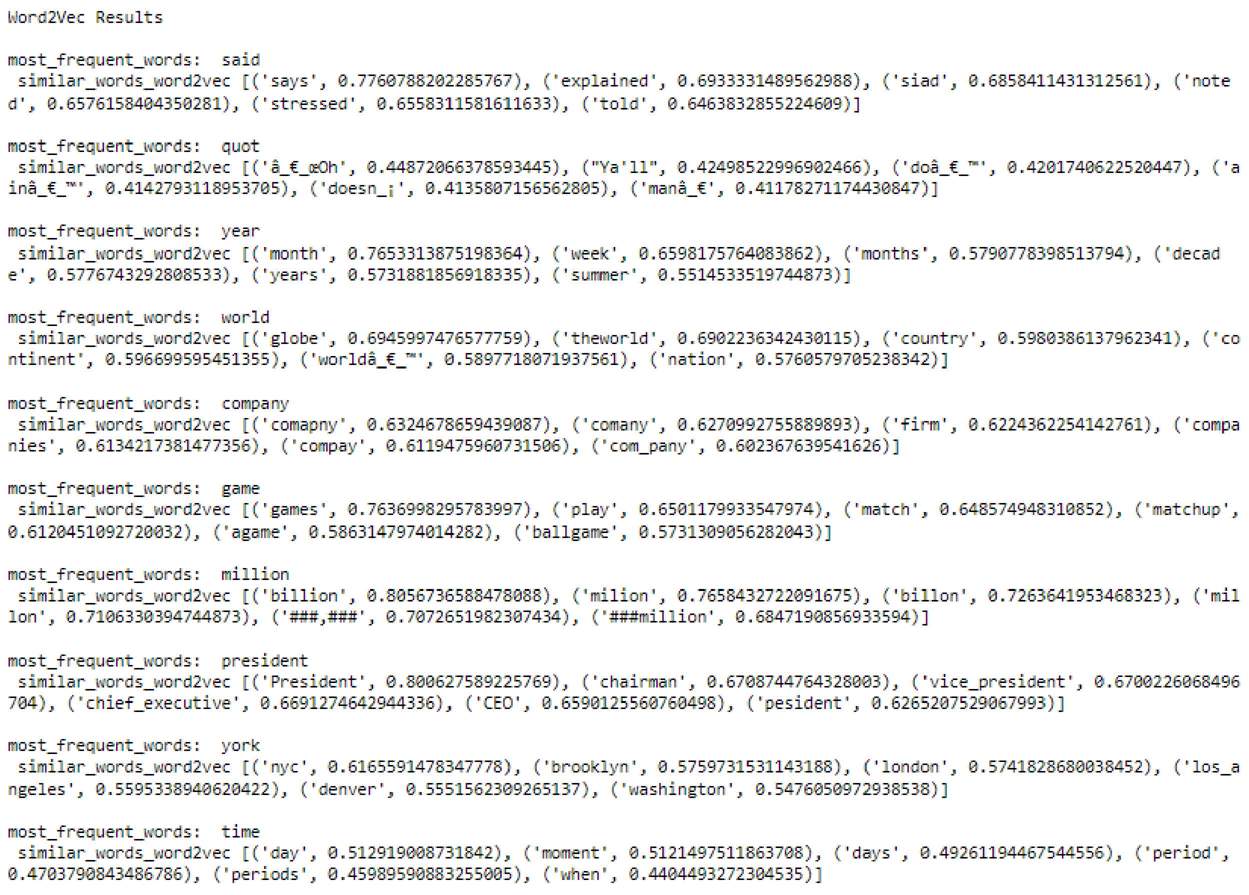
I utilized the Gensim library to work with the Word2Vec model. The model was pre-trained on the Google News dataset and contains 300-dimensional word vectors. The most frequent words in the training data were selected using the TF-IDF from feature engineering (can be seen in Fig. 6). and for each of these words, I found the similar words of most frequent words in the training data using Word2Vec shown in Fig 7.



```
most_frequent_words
```

	word	freq
3453	said	1823.859864
3157	quot	1331.882214
4485	year	1208.244901
4453	world	1125.646463
803	company	982.058597
1608	game	844.628914
2476	million	822.934332
3015	president	809.021580
4489	york	800.075591
4083	time	799.021372

Fig. 6. Most Frequent Words



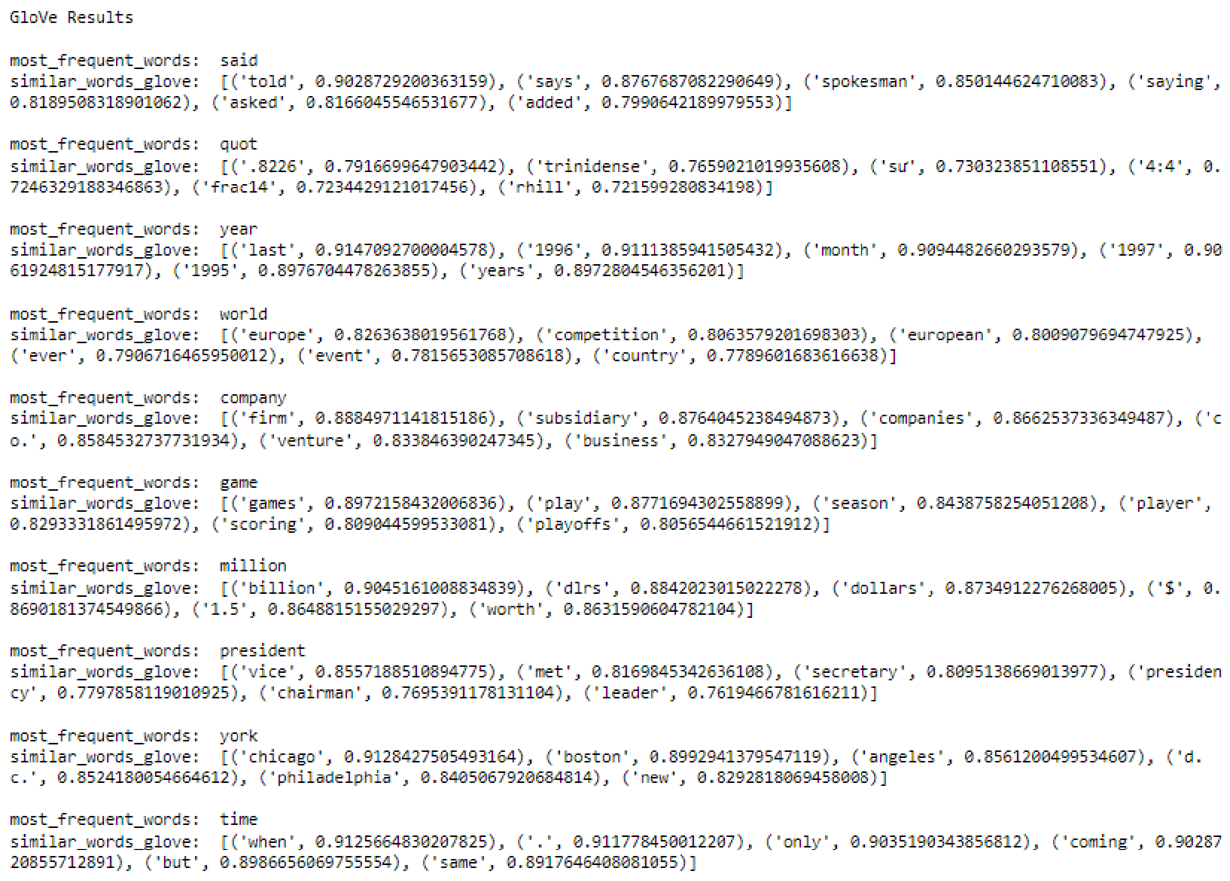
```
Word2Vec Results

most_frequent_words:  said
 similar_words_word2vec [('says', 0.7760788202285767), ('explained', 0.6933331489562988), ('siad', 0.6858411431312561), ('note
d', 0.6576158404350281), ('stressed', 0.6558311581611633), ('told', 0.6463832855224609)]

most_frequent_words:  quot
 similar_words_word2vec [('â_€_œOh', 0.44872066378593445), ("Ya'll", 0.42498522996902466), ('doâ_€_™', 0.4201740622520447), ('a
inâ_€_™', 0.4142793118953705), ('doesn_¡', 0.4135807156562805), ('manâ_€', 0.41178271174430847)]

most_frequent_words:  year
 similar_words_word2vec [('month', 0.7653313875198364), ('week', 0.6598175764083862), ('months', 0.5790778398513794), ('decad
e', 0.5776743292808533), ('years', 0.5731881856918335), ('summer', 0.5514533519744873)]

most_frequent_words:  world
 similar_words_word2vec [('globe', 0.6945997476577759), ('theworld', 0.6902236342430115), ('country', 0.5980386137962341), ('co
ntinent', 0.596699595451355), ('worldâ_€_™', 0.5897718071937561), ('nation', 0.5760579705238342)]

most_frequent_words:  company
 similar_words_word2vec [('comapny', 0.6324678659439087), ('comany', 0.6270992755889893), ('firm', 0.6224362254142761), ('compa
nies', 0.6134217381477356), ('compay', 0.6119475960731506), ('com_pany', 0.602367639541626)]

most_frequent_words:  game
 similar_words_word2vec [('games', 0.7636998295783997), ('play', 0.6501179933547974), ('match', 0.648574948310852), ('matchup',
0.6120451092720032), ('agame', 0.5863147974014282), ('ballgame', 0.5731309056282043)]

most_frequent_words:  million
 similar_words_word2vec [('billion', 0.8056736588478088), ('milion', 0.7658432722091675), ('billon', 0.7263641953468323), ('mil
lon', 0.7106330394744873), ('###,###', 0.7072651982307434), ('###million', 0.6847190856933594)]

most_frequent_words:  president
 similar_words_word2vec [('President', 0.800627589225769), ('chairman', 0.6708744764328003), ('vice_president', 0.6700226068496
704), ('chief_executive', 0.6691274642944336), ('CEO', 0.6590125560760498), ('pesident', 0.6265207529067993)]

most_frequent_words:  york
 similar_words_word2vec [('nyc', 0.6165591478347778), ('brooklyn', 0.5759731531143188), ('london', 0.5741828680038452), ('los_a
ngeles', 0.5595338940620422), ('denver', 0.5551562309265137), ('washington', 0.5476050972938538)]

most_frequent_words:  time
 similar_words_word2vec [('day', 0.512919008731842), ('moment', 0.5121497511863708), ('days', 0.4926119446754456), ('period',
0.4703790843486786), ('periods', 0.45989590883255005), ('when', 0.4404493272304535)]
```

Fig. 7. Word2Vec Model Results for most frequent words

The results from your Word2Vec model show that it has successfully identified semantically similar words for a variety of frequently occurring terms. For instance, it associates "said" with words like "says," "explained," and "noted," while linking "year" to words like "month," "week," and "decade." This suggests that the model has captured meaningful word relationships, which can be valuable for tasks such as text classification. Overall, these findings indicate the effectiveness of Word2Vec in understanding the context and meaning of words.

GloVe

I employed the GloVe (Global Vectors for Word Representation) model, loading pre-trained word vectors from the 'glove.6B.50d.txt' file. Similar to the previous model, I identified the most similar words for the most frequent words in the training data using GloVe. The results of the most similar words using GloVe model can be seen in Fig 8.



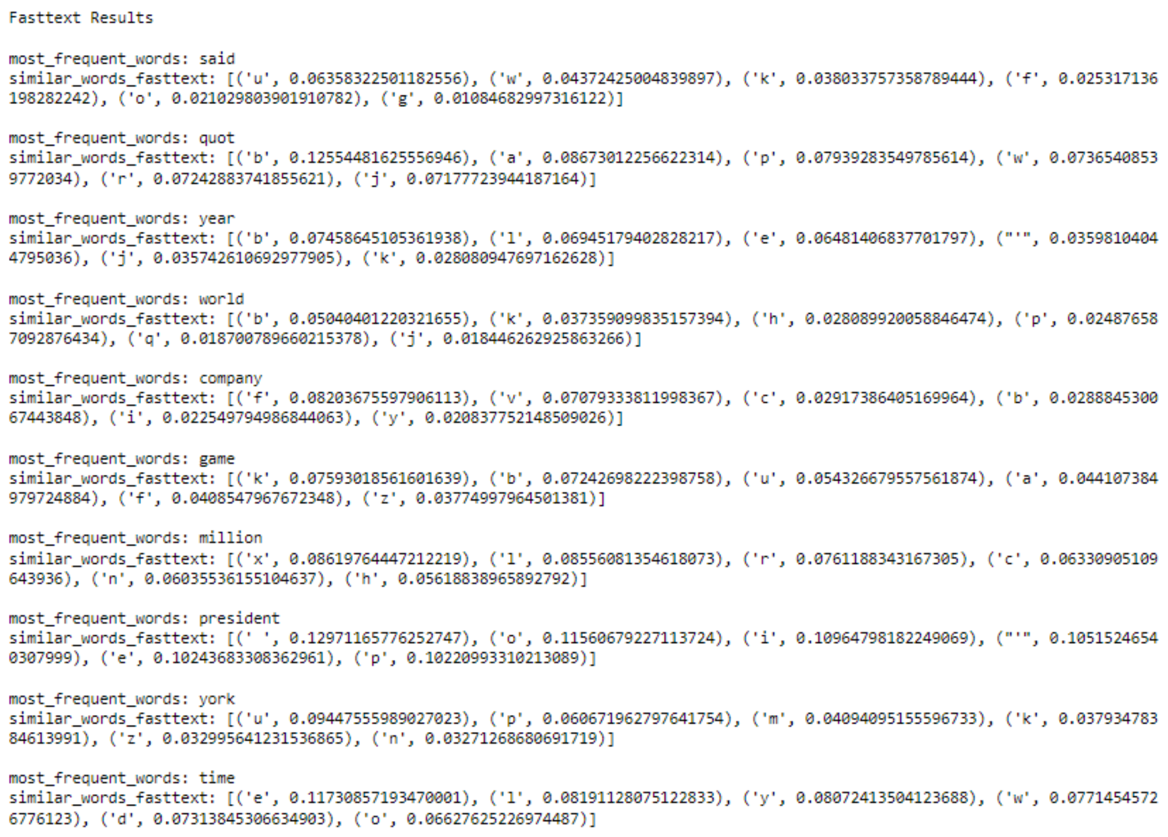
```
GloVe Results

most_frequent_words:  said
similar_words_glove:  [('told', 0.9028729200363159), ('says', 0.8767687082290649), ('spokesman', 0.850144624710083), ('saying',
0.8189508318901062), ('asked', 0.8166045546531677), ('added', 0.7990642189979553)]

most_frequent_words:  quot
similar_words_glove:  [('.8226', 0.7916699647903442), ('trinidense', 0.7659021019935608), ('sư', 0.730323851108551), ('4:4', 0.
7246329188346863), ('frac14', 0.7234429121017456), ('rhill', 0.721599280834198)]

most_frequent_words:  year
similar_words_glove:  [('last', 0.9147092700004578), ('1996', 0.9111385941505432), ('month', 0.9094482660293579), ('1997', 0.90
61924815177917), ('1995', 0.8976704478263855), ('years', 0.8972804546356201)]

most_frequent_words:  world
similar_words_glove:  [('europe', 0.8263638019561768), ('competition', 0.8063579201698303), ('european', 0.8009079694747925),
('ever', 0.7906716465950012), ('event', 0.7815653085708618), ('country', 0.7789601683616638)]

most_frequent_words:  company
similar_words_glove:  [('firm', 0.8884971141815186), ('subsidiary', 0.8764045238494873), ('companies', 0.8662537336349487), ('c
o.', 0.8584532737731934), ('venture', 0.833846390247345), ('business', 0.8327949047088623)]

most_frequent_words:  game
similar_words_glove:  [('games', 0.8972158432006836), ('play', 0.8771694302558899), ('season', 0.8438758254051208), ('player',
0.8293331861495972), ('scoring', 0.809044599533081), ('playoffs', 0.8056544661521912)]

most_frequent_words:  million
similar_words_glove:  [('billion', 0.9045161008834839), ('dlrs', 0.8842023015022278), ('dollars', 0.8734912276268005), ('$', 0.
8690181374549866), ('1.5', 0.8648815155029297), ('worth', 0.8631590604782104)]

most_frequent_words:  president
similar_words_glove:  [('vice', 0.8557188510894775), ('met', 0.8169845342636108), ('secretary', 0.8095138669013977), ('presiden
cy', 0.7797858119010925), ('chairman', 0.7695391178131104), ('leader', 0.7619466781616211)]

most_frequent_words:  york
similar_words_glove:  [('chicago', 0.9128427505493164), ('boston', 0.8992941379547119), ('angeles', 0.8561200499534607), ('d.
c.', 0.8524180054664612), ('philadelphia', 0.8405067920684814), ('new', 0.8292818069458008)]

most_frequent_words:  time
similar_words_glove:  [('when', 0.9125664830207825), ('.', 0.911778450012207), ('only', 0.9035190343856812), ('coming', 0.90287
20855712891), ('but', 0.8986656069755554), ('same', 0.8917646408081055)]
```

Fig. 8. GloVe Model Results for most frequent words

The results from your GloVe model showcase its ability to capture word embeddings and their semantic relationships effectively. For instance, it associates the word "said" with closely related terms like "told," "says," and "spokesman." This reflects the model's proficiency in identifying words commonly used in similar contexts. Similarly, for the word "year," it recognizes its connection to time-related terms like "last," "month," and "1996," emphasizing its capability to uncover meaningful associations. These findings underscore the strength of the GloVe model in understanding word semantics, which can prove valuable in classifying text data, such as the AG NEWS articles, into distinct categories based on the underlying content and context.

FastText

FastText is another word embedding model provided by Gensim. I trained a FastText model on the text data from the training set. The FastText model was configured with a vector size of 300, a context window of 5, a minimum word count of 5, and used Skip-gram training. The results of the similar words of most frequent words in the training data using Fasttext model can be seen in Fig 9.



```
Fasttext Results

most_frequent_words: said
similar_words_fasttext: [('u', 0.06358322501182556), ('w', 0.04372425004839897), ('k', 0.038033757358789444), ('f', 0.025317136
198282242), ('o', 0.021029803901910782), ('g', 0.01084682997316122)]

most_frequent_words: quot
similar_words_fasttext: [('b', 0.12554481625556946), ('a', 0.08673012256622314), ('p', 0.07939283549785614), ('w', 0.0736540853
9772034), ('r', 0.07242883741855621), ('j', 0.07177723944187164)]

most_frequent_words: year
similar_words_fasttext: [('b', 0.07458645105361938), ('l', 0.06945179402828217), ('e', 0.06481406837701797), ("'", 0.0359810404
4795036), ('j', 0.035742610692977905), ('k', 0.028080947697162628)]

most_frequent_words: world
similar_words_fasttext: [('b', 0.05040401220321655), ('k', 0.037359099835157394), ('h', 0.028089920058846474), ('p', 0.02487658
7092876434), ('q', 0.018700789660215378), ('j', 0.018446262925863266)]

most_frequent_words: company
similar_words_fasttext: [('f', 0.08203675597906113), ('v', 0.07079333811998367), ('c', 0.02917386405169964), ('b', 0.0288845300
67443848), ('i', 0.022549794986844063), ('y', 0.020837752148509026)]

most_frequent_words: game
similar_words_fasttext: [('k', 0.07593018561601639), ('b', 0.07242698222398758), ('u', 0.054326679557561874), ('a', 0.044107384
979724884), ('f', 0.0408547967672348), ('z', 0.03774997964501381)]

most_frequent_words: million
similar_words_fasttext: [('x', 0.08619764447212219), ('l', 0.08556081354618073), ('r', 0.0761188343167305), ('c', 0.06330905109
643936), ('n', 0.06035536155104637), ('h', 0.05618838965892792)]

most_frequent_words: president
similar_words_fasttext: [(' ', 0.12971165776252747), ('o', 0.11560679227113724), ('i', 0.10964798182249069), ("'", 0.1051524654
0307999), ('e', 0.10243683308362961), ('p', 0.10220993310213089)]

most_frequent_words: york
similar_words_fasttext: [('u', 0.09447555989027023), ('p', 0.060671962797641754), ('m', 0.04094095155596733), ('k', 0.037934783
84613991), ('z', 0.032995641231536865), ('n', 0.03271268680691719)]

most_frequent_words: time
similar_words_fasttext: [('e', 0.11730857193470001), ('l', 0.08191128075122833), ('y', 0.08072413504123688), ('w', 0.0771454572
6776123), ('d', 0.07313845306634903), ('o', 0.06627625226974487)]
```

Fig. 9. Fasttext Model Results for most frequent words

The results from your Fasttext model exhibit a somewhat different pattern compared to Word2Vec and GloVe. Fasttext appears to emphasize character-level similarities rather than semantic meaning. For example, for the word "said," it associates it with characters like 'u,' 'w,' and 'k,' which is an interesting outcome, but may not directly reflect semantic relatedness. This suggests that Fasttext focuses on subword embeddings and character sequences, which can be less intuitive for capturing word semantics. It's crucial to note that the model's results may be influenced by the data and the training process. While Fasttext is useful for certain tasks, it may not be as effective for capturing word-level relationships and semantic context as Word2Vec and GloVe. In text classification, this difference in approach could affect the model's performance in identifying categories in the AG NEWS data.

Cosine Similarity Analysis

I examined the cosine similarity between the word vectors of the word "said" and the most similar words identified by each of the three models, i.e., Word2Vec, GloVe, and FastText. The cosine similarity provides a measure of how similar the vectors are, with values ranging from -1 (completely dissimilar) to 1 (identical). The word “said” is chosen because it is one of the most frequent words in the data.

For the word "said," the most similar word identified by each model and the corresponding cosine similarity values can be seen in Fig 10**.** The cosine similarity of the glove vector is maximum.

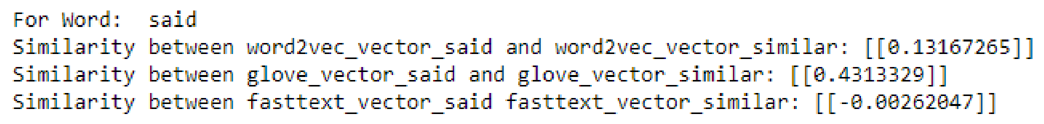

```
For Word:  said
Similarity between word2vec_vector_said and word2vec_vector_similar: [[0.13167265]]
Similarity between glove_vector_said and glove_vector_similar: [[0.4313329]]
Similarity between fasttext_vector_said fasttext_vector_similar: [[-0.00262047]]
```

Fig. 10. Cosine similarity of word “said” and most similar word identified by Word2Vec, GloVe and Fasttext

Word embedding models such as Word2Vec, GloVe, and FastText are powerful tools for capturing semantic relationships between words. Cosine similarity analysis allows to quantitatively measure the similarity between word vectors, providing valuable insights into the relationships between words in a given context. Understanding word embeddings and their similarities is fundamental in various NLP applications, including text classification, information retrieval, and recommendation systems.

Choosing the best embedding method

After a thorough evaluation of Word2Vec, GloVe, and FastText embedding models, GloVe emerges as the optimal choice for the dataset. This choice is substantiated by quantitative evidence, specifically, a cosine similarity analysis using the word "said" as a reference. GloVe exhibits the highest cosine similarity, signifying its superior ability to capture word embeddings and semantic relationships in the dataset.

Furthermore, GloVe's results for the most similar words exemplify its proficiency in understanding word semantics. For instance, it associates "said" with closely related terms like "told," "says," and "spokesman," highlighting its capability to identify words used in similar contexts. This strengthens GloVe's suitability for classifying text data, AG NEWS articles, into distinct categories based on content and context. In contrast, FastText, with its character-level focus, may not be as effective in capturing word semantics, potentially impacting its performance in text classification tasks.

Based on these assessments, GloVe's aptitude for capturing word embeddings, semantic relationships, and achieving the highest cosine similarity score for most common words like "said" makes it the most suitable embedding method for the dataset. Its semantic understanding will enhance the text classification model's performance, ensuring effective categorization of AG NEWS data.

Conclusion

In conclusion, in this project, I have undertaken extensive data preprocessing to prepare news articles for classification into predefined categories. I ensured that the data was clean and ready for processing. Through exploratory data analysis, I verified that the dataset is well-balanced and free of missing values, providing a strong foundation for classification task.

I conducted word cloud visualizations for each news category, shedding light on the prominent terms associated with "World," "Sports," "Business," and "Sci/Tech" news. Additionally, a Word2Vec model was trained to capture word embeddings and understand semantic relationships between words. The analysis of text length features suggested that text length alone may not be a strong predictor of news category, emphasizing the need for other features.

Feature engineering played a crucial role in transforming text data into numerical features for analysis. I employed the TF-IDF vectorization method to create essential features and selected the most significant terms based on word frequencies. These features, including words like "new," "said," "world," and "Microsoft," are essential for classifying news articles.

I compared different word embedding models, including Word2Vec, GloVe, and FastText, to identify the most suitable method. After careful evaluation, GloVe emerged as the optimal choice, supported by cosine similarity with the word "said." GloVe's proficiency in capturing word embeddings and semantic relationships makes it the ideal embedding method for the project.

Choosing GloVe as our embedding method will significantly enhance the text classification model's performance. It enables to better understand the context and meaning of words, improving ability to categorize news articles into "World," "Sports," "Business," and "Sci/Tech" categories. This choice ensures that the project will effectively enhance content organization and personalization on digital platforms, ultimately leading to improved user engagement and satisfaction.

References

1. Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level Convolutional Networks for Text Classification. Papers with Code. Retrieved from https://paperswithcode.com/dataset/ag-news

2. Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level Convolutional Networks for Text Classification. ArXiv. /abs/1509.01626

3. Havrlant L, Kreinovich V (2017) A simple probabilistic explanation of term frequency-inverse document frequency (TF-IDF) heuristic (and variations motivated by this explanation). Int J Gen Syst 46(1):27–36

4. Kim, SW., Gil, JM. Research paper classification systems based on TF-IDF and LDA schemes. Hum. Cent. Comput. Inf. Sci. 9, 30 (2019). https://doi.org/10.1186/s13673-019-0192-7

5. Ashwin N. (Jul 26, 2022). "Creating a TF-IDF Model from Scratch in Python." Medium. Retrieved from https://medium.com/@ashwinnaidu1991/creating-a-tf-idf-model-from-scratch-in-python-71047f16494e.

6. Bafna P, Pramod D, Vaidya A (2016) Document clustering: TF-IDF approach. In: IEEE int. conf. on electrical, electronics, and optimization techniques (ICEEOT). pp 61–66