TEXT MINING PROJECT:

NERC, Sentiment Analysis, and Topic Modeling

Group [redacted]

This study implements and compares multiple natural language processing approaches across three tasks: Named Entity Recognition and Classification (NERC), sentiment analysis, and topic modeling. For NERC, spaCy's statistical NERC and a BERT-based model (dslim/bert-base-NER) were evaluated and compared based on token-level accuracy and entity span performance. Sentiment analysis was done using supervised a suoervised Logistic Regression classifier and the rule-based VADER lexicon to assess machine learning versus lexicon-based approaches. Topic modeling employed three methods for document classification: supervised Logistic Regression on TF-IDF features, hybrid LDA and Logistic Regression, via latent feature extraction and utilizing as features for Log. Regression, and a hybrid LSA and Logistic Regression model with a similar idea but with CountVectorizer features.

01 NERC ANALYSIS

THE DATA USED FOR THE TASK

The task in NERC, is to recognize named entities in the text and classify them according to their types. A manually annotated test set was used for the analysis. In this dataset, sentences were built up to have tokens and associated BIO tags next to them. The gold labels denoted a variety of entity types, such as the PERSON, ORG, and LOCATION categories. The *load_test_data()* function, which was put in place to guarantee appropriate data structuring, made the data loading procedure easier.

MODEL IMPLEMENTATION

spaCy Model: The first model implemented was the spaCy NERC model, which was integrated through the *apply_spacy_ner()* function. SpaCy processes text via a transition-based parser trained on OntoNotes [1]. In this implementation, sentences were processed on a token-by-token basis, where BIO tags were directly generated. Entity type information was preserved in the tags, resulting in formats such as "B-PERSON" and "I-ORG".

BERT Model: The second model utilized was the BERT NERC model, implemented through the *apply_bert_ner()* function. generates contextualized token embeddings that we convert to BIO tags via custom logic to strip "##" artifacts and align subword spans to original tokens [2].This implementation was based on the Hugging Face pipeline, where special attention was paid to subword token handling. The '##' artifacts were removed during processing, and entity predictions were carefully mapped to the BIO format. A specific logic was developed for matching entity spans to individual tokens.

The implementation process was initiated with data loading through the initialization module, followed by the loading of pre-trained models via *load_pretrained_models()*. Both models were then applied to process the test sentences, with results being structured uniformly to include tokens and their corresponding NERC tags. This entire process was encapsulated within the main *models.py* module script. The analysis phase was conducted through three main components: a comparison of model outputs through the *compare()* function, entity extraction via the *extract()* function, and a detailed analysis of results using the *analysis()* function.

Motivation for Pretrained Usage:

- The decision to use a pretrained model without fine-tuning was motivated by the robustness of CoNLL-2003 fine-tuned models and time constraints.
- Pretrained transformer-based models such as dslim/bert-base-NER perform well out of the box on many entity types, especially those covered by their training corpus.

DISCUSSION AND RESULTS

Split-up Multiword Entities:

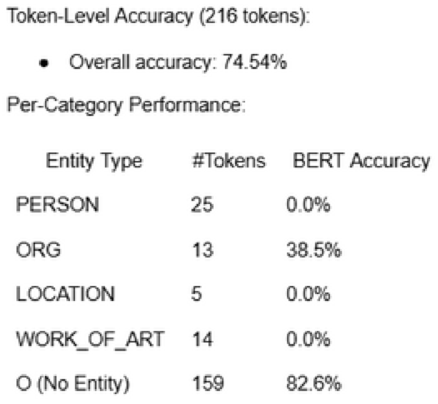
- Multi-token names such as "Pharoah Sanders" and "Kieran Culkin" were often split, with only partial recognition (e.g., only "Culkin" tagged as PER).

Category Misclassifications:

- Creative works were often labeled as MISC or missed entirely. For example, "Mona Lisa" was labeled incorrectly as MISC.

False Negatives:

- Several expected entities (especially LOC and WORK_OF_ART) were missed altogether, likely due to domain mismatch between training data and the test set.



Token-Level Accuracy (216 tokens):

- Overall accuracy: 74.54%

Per-Category Performance:

Entity Type	#Tokens	BERT Accuracy
PERSON	25	0.0%
ORG	13	38.5%
LOCATION	5	0.0%
WORK_OF_ART	14	0.0%
O (No Entity)	159	82.6%

Fig.1 Token-Level Accuracy and Per-Category Performance

Example Sentence: "If you're visiting Paris, make sure to see the Louvre, as they exhibit the Mona Lisa!"

- True entities: [(Paris, LOCATION), (Louvre, ORG), (Mona Lisa, WORK_OF_ART)]
- Predicted: [(Paris, LOC), (Louvre, ORG), (Mona, MISC)] → Multiple partial or incorrect predictions

CONCLUSIONS

Through this implementation, a comprehensive comparison between traditional NLP (spaCy) and transformer-based (BERT) approaches to NERC was achieved. The spaCy model showed strength in maintaining entity coherence through its direct BIO tagging system, where entity boundaries were precisely defined through the token-by-token processing approach. This became clear where the B-I-O transitions were explicitly managed within the *apply_spacy_ner()* function. The BERT-based implementation, processed through the *apply_bert_ner()* function, demonstrated contextual understanding, even though it had some trouble in entity span alignment. The implemented solution for handling BERT's subword tokenization had some issues, particularly the removal of '##' artifacts and the span-to-token alignment mechanism. However, this approach occasionally resulted in fragmented (split-up) entity recognition, especially in cases of complex multi-token entities.

Three key limitations were identified within this implementation: inconsistent entity type mapping between models, conversion challenges from span-level to token-level BIO tags, and varying performance across entity types. Despite these constraints, the modular architecture proved effective, enabling direct comparison within a unified framework. The insights gained suggest that optimal NERC performance could be achieved through a further fine-tuning of the pre-trained models, with the current architecture well-positioned for future enhancements. The models could be improved through the implementation of context-specific datasets that teaches on top of what the pre-trained model already knows.The current implementation serves as a robust foundation for named entity recognition and classification tasks, where two distinct approaches were successfully integrated and compared. The modular nature of the full pipeline implementation ensures that future improvements and extensions can be readily incorporated into the existing framework.

02 SENTIMENT ANALYSIS

MODEL IMPLEMENTATION

The task of sentiment analysis is the examination of subjectivity expressed within textual data [3]. This section is concerned with sentiment polarity classification, motivated by a small test dataset with 18 sentence instances labeled with 'positive', 'neutral', and 'negative'. Two methodological approaches are selected to achieve this task: a simple rule-based approach with the VADER (Valence Aware Dictionary for sEntiment Reasoning) model [4] and a machine learning-based approach with a Logistic Regression algorithm from the *scikit-learn* library [5]. The modela are compared with each other to find the best-suited model for the task. These models are also selected due to their architecture being suited for polarity classification.

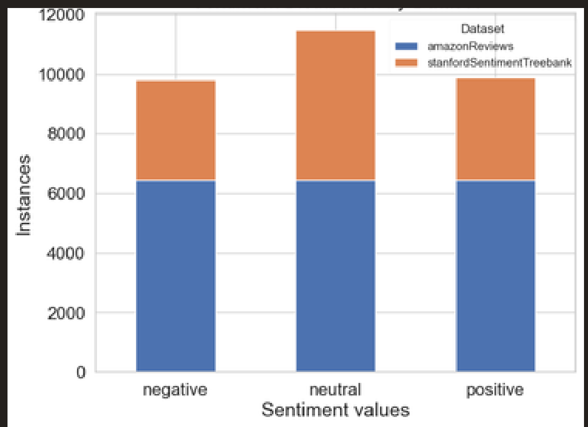
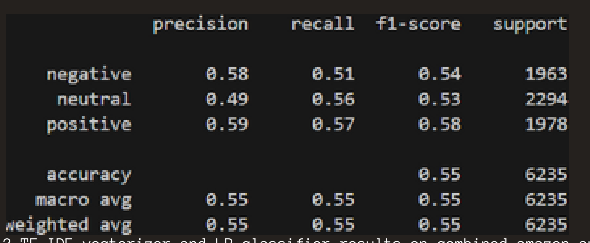


Fig.2 combined amazon and stanford dataset label distribution

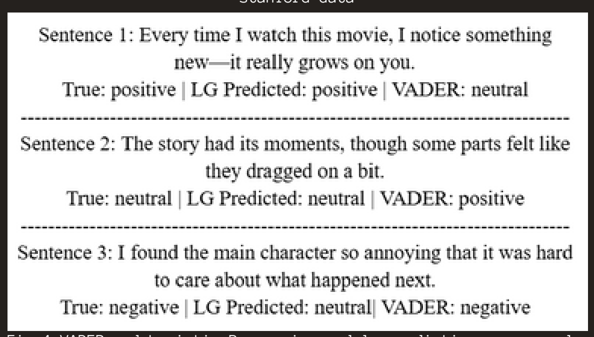


```
              precision    recall  f1-score   support

    negative       0.58      0.51      0.54      1963
     neutral       0.49      0.56      0.53      2294
    positive       0.59      0.57      0.58      1978

    accuracy                           0.55      6235
   macro avg       0.55      0.55      0.55      6235
weighted avg       0.55      0.55      0.55      6235
```

Fig.3 TF-IDF vectorizer and LR classifier results on combined amazon and stanford data



Sentence 1: Every time I watch this movie, I notice something new—it really grows on you.
True: positive | LG Predicted: positive | VADER: neutral

Sentence 2: The story had its moments, though some parts felt like they dragged on a bit.
True: neutral | LG Predicted: neutral | VADER: positive

Sentence 3: I found the main character so annoying that it was hard to care about what happened next.
True: negative | LG Predicted: neutral| VADER: negative

Fig.4 VADER and Logistic Regression models predictions on example sentences from the test set

DATASET COMBINED

Data search was conducted with the data needed for test data classification in mind. Two datasets were found and combined together: Amazon Reviews 2023 from McAuley Lab [6] and movie reviews from SST (Stanford Sentiment Treebank) [7]. To prevent class imbalance, the Amazon reviews dataset was cropped to match the least occurring sentiment label. The datasets were then combined and shuffled together into a CSV file. (this can be found in the *sentiment-data-preprocessing.ipynb* file)

RESULTS

The most efficient model was found to be Logistic Regression with TF-IDF (term frequency - inverse document frequency). The model was cross-validated and a 0.55 train accuracy score was obtained from the combined dataset.

Looking at the classification results in the figure, the (macro-averaged) F1-score of the positive class is higher than the negative and neutral, indicating the model handles positive classes slightly better than negative and neutral. Support indicates that class imbalance is likely not an issue.

Then the Logistic Regression model and VADER were both evaluated on the test data, resulting in 0.33 and 0.56 accuracies respectively. VADER's accuracy score indicates that it is no better than random guessing.

For the first example sentence, VADER assigned 'neutral' likely due to the sentence not having enough positive valence-scored words in it. It fails to capture the phrase 'grows on you' as Logistic Regression does by learning more complex contextual and higher-level patterns. The second sentence follows the same pattern. The Logistic Regression model captures the mixed sentiments of 'had its moments' and 'dragged on', whereas VADER assigns positive sentiment likely due to phrases like 'had its moments'.

CONCLUSION

This sentiment analysis compared rule-based and machine learning approaches using a combined dataset from Amazon Reviews 2023 and Stanford Sentiment Treebank. While the Logistic Regression model with TF-IDF vectorization achieved a higher accuracy level on the test set, both approaches have their own limitations: VADER excels at explicit sentiment words but struggles with contextual phrases, while Logistic Regression better captures complex contextual phrases but can misinterpret mixed sentiments. Low test accuracies suggest that sentiment analysis remains a challenging task, indicating that more advanced models and perhaps datasets of bigger size could be utilized to improve performance.

03 TOPIC MODELING

MODEL IMPLEMENTATION

The task of topic modeling is defined as the creation of models that can decide what topic a given text/document/collection is, based on some training.

Datasets & Preprocessing: The first dataset [7] is a movie review dataset, with multiple different features. The main focus in this task is identifying texts written about movie reviews, so the features other than the main text does not matter much. In the preprocessing step of this dataset, the texts are filtered to only be longer than 25 characters. Only the first 175 instances of text are split into sentences and labeled 'movie', since the other datasets have fewer instances/sentences. The second dataset [8] contains book reviews from Amazon, with features containing book titles, texts, etc. In the preprocessing step, a text is split into sentences only if it is about a unique book, to ensure diverse vocabulary. A random sentence from the splitting of the text is then selected, and this process is only done for the first 16 thousand instances, since there are lots of reviews about the same books. The third dataset, GOAL, [9], and the fourth Irish Times [10], are datasets containing texts focused around sports. The Irish Times dataset contains many news headlines, but only the sports-related ones are extracted. Goal contains text versions of live-recorded football commentation. Goal and Irish Times both contain fewer text than the first two datasets, and hence are combined. This also enables more diverse sport-related speech, since Irish Times' texts are more formal and Goal's more informal examples of language use.

After the four datasets' texts have been extracted, they are labeled to be one of the three target classes for the test set, and then combined to be one dataset. The final preprocessing step appears at the NLP tokenization, stop word removal, lemmatization, and lowercasing. The returned dataset from the preprocessing module is not stored internally, but rather used right away as a Python list of tuple objects containing the text and the appended label.

Logistic Regression Model: The supervised logistic regression model utilizes TF-IDF vectorization. The vectorizer is configured with English stop word removal and a maximum feature limit of 10 thousand to add richness. The preprocessed text data is then converted into numerical representations. For the validation set, a 5-fold stratified cross-validation approach is employed to ensure balanced representation of all three classes (sports, movies, books) across training and testing splits (not 10, since the combined dataset is only ~30k instances) [11]. Within each fold, a basic logistic regression classifier is defined with a maximum iteration limit of 1000 (the default 100 in *sklearn* isn't enough to handle 10k *max_features* from the TF-IDF), and trained on the vectorized training data. The model's performance is evaluated on each test fold, with predictions collected across all folds for mean validation accuracy analysis. The implementation also includes a *predict()* function that utilizes the trained vectorizer and the classifier to make predictions on new text instances [12].

LDA: The implementation contains a supervised approach to LDA, as inspired by [13, 14]. This is achieved by combining LDA with a Logistic Regression classifier (instead of sLDA). The LDA model employs a probabilistic generative approach to discover hidden topics in the document collection [15]. LDA assumes that each document is a mixture of topics, and each topic is characterized by a distribution over words [16]. First, the three target labels are turned into numerical values 0, 1, 2 with *sklearn's LabelEncoder*. The implementation utilizes a CountVectorizer rather than TF-IDF, as LDA operates on raw word counts to model the probabilistic word-topic relationships effectively (meaning topics come from co-occurrance) [17]. The model is configured with 10 latent topics by default, each document is transformed into a topic probability distribution, where the resulting feature vector represents the likelihood of the document belonging to each discovered topic. The latent topics acquired from LDA are fit to a logistic regression model, which learns from these features to classify into the three target labels in the dataset. The model contains a *predict* function within.

LSA: The LSA model applies singular value decomposition through TruncatedSVD to reduce the dimensionality of TF-IDF weighted term-document matrices. LSA operates under the assumption that words with similar meanings will occur in similar contexts, using linear algebraic techniques to uncover latent semantic relationships [15, 18, 19]. The choice of TF-IDF comes from the weighting helping emphasize discriminative terms while reducing the influence of common words (since TF-IDF shows how important a word is in a collection of documents). First, the labels are converted into numerical values via *LabelEncoder*. Then the TruncatedSVD (the key LSA component) is fit to the data to reduce dimensionality, and combined with a Logistic Regression classifier, again with *max_iterations* set to 1000 [12]. The model contains a nested *predict()* function. Both LDA and LSA models implement identical cross-validation frameworks with 5-fold stratified splitting just like the Logistic Regression model.

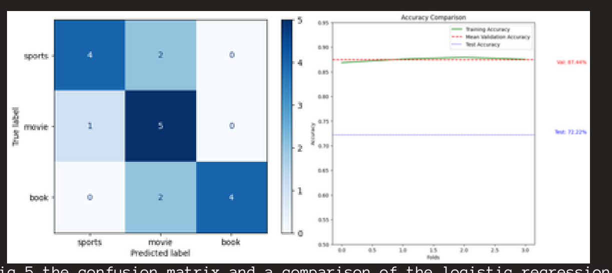


Fig.5 the confusion matrix and a comparison of the logistic regression

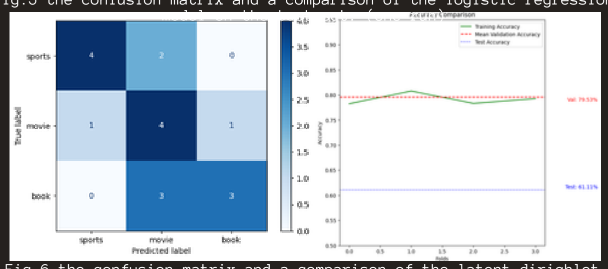


Fig.6 the confusion matrix and a comparison of the latent dirichlet allocation model on the test set. (one run)

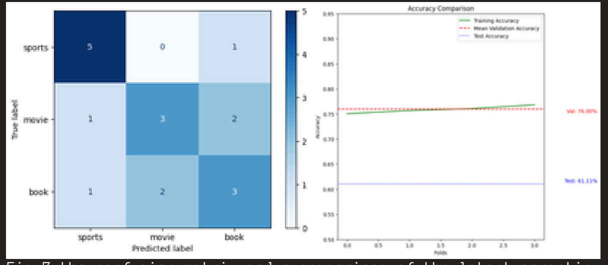
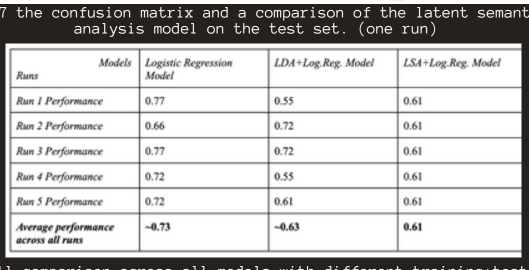


Fig.7 the confusion matrix and a comparison of the latent semantic analysis model on the test set. (one run)



Models Runs	Logistic Regression Model	LDA+Log.Reg. Model	LSA+Log.Reg. Model
Run 1 Performance	0.77	0.55	0.61
Run 2 Performance	0.66	0.72	0.61
Run 3 Performance	0.77	0.72	0.61
Run 4 Performance	0.72	0.55	0.61
Run 5 Performance	0.72	0.61	0.61
Average performance across all runs	**~0.73**	**~0.63**	**0.61**

Fig.8 full comparison across all models with different training+test runs

ANALYSIS

Logistic Regression Model:

- *Training Mean Accuracy:* 87.34%
- *Test Accuracy:* 72.22% (5 incorrect out of 18)

The false positives in the 'movie' class were the most common error, misclassifying two book reviews and two sports-related sentences. One sentence about sports strategy was interpreted as a movie review (e.g., "Both teams are playing it safe…"), likely due to lacking domain-specific language. This could have been the case for other sentences in the training phase as well. The errors suggest the model captures structural and lexical features (sentiment, syntax, etc.) more strongly than true thematic content, a known limitation of TF-IDF + Logistic Regression pipelines in topic tasks [20]. However, the model performed well when domain-specific vocabulary was clear. The model also showed no obvious class imbalance, meaning the preprocessing phase was balanced & a success.

LDA Model:

- *Training Mean Accuracy:* 79.53%, *Test Accuracy:* 61.11% (7 incorrect out of 18)

Most Common Error: False positives in the 'book' class, misclassified as 'movie' (3 out of 7 errors), indicating difficulty distinguishing literary from cinematic language. The model also made 2 errors on 'sports', predicting 'movie' instead, often when the 'sports' sentence had an emotional or generic structure (e.g., "disappointing finish" or "safe play"). Only one true 'movie' was misclassified, showing relatively strong performance in the class. The model performs well on predictions when domain-specific terms are present, however, LDA's unsupervised nature makes it difficult to align latent topics with fine-grained class distinctions. The model likely inferred topics based on shared style or structure rather than topic-domain identity.

LSA Model:

Training Mean Accuracy: 76%, Test Accuracy: 61.11% (7 incorrect out of 18)

Most Common Error: The LSA model made diverse types of misclassifications, where 3 errors were book texts misclassified as 'movie' or 'sports' and 3 errors were movie texts misclassified as 'book' or 'sports'. Only 1 sports text was misclassified as 'book', likely due to more reflective or emotional language rather than technical sports vocabulary. The sports class performance of the LSA, however, seems to be much better than the other two labels. The model only made one false positive prediction for the sports class, namely 'book'. This shows more balanced confusion compared to the other models.

CONCLUSIONS, LIMITATIONS & FUTURE IMPROVEMENTS

The results show that the Logistic Regression is generally better than the other two implementations, whereas LDA and LSA have a very slight difference in performance. LDA shows to be unstable & stochastic, getting different test accuracies through different training+testing runs, whereas the LSA shows to have small standard deviation but is more deterministic than LDA. Since Logistic Regression doesn't model topics explicitly, just raw features and class correlations, it can't interpret/learn 'topics', but rather just how to differentiate the text. This approach still performs better than LDA and LSA, models created for Topic Modeling specifically, most likely because the dataset size is small. LDA, being unsupervised, may produce (latent) topics that do not align with the actual labels, leading to noisy features for the Logistic Regression classifier model. LSA, while effective in dimensionality reduction, discards supervised information and offers limited interpretability. Moreover, while LSA and LDA both achieve similar accuracy, they make different types of errors: LDA skews heavily toward one label (e.g., 'movie'), while LSA shows a broader distribution of misclassifications. The training phase and evaluation of LSA is much faster compared to LDA, and in such small (supervised) datasets, LSA might be the better option. All three models rely on bag-of-words representations, ignoring context and word order. In future work, integrating supervised topic models (e.g., sLDA), fine-tuning dimensionality reduction with label-aware methods, or replacing TF-IDF with contextual embeddings from transformers like BERT could improve classification performance [21]. Larger and more balanced datasets would also help mitigate current generalization limitations.

Group Contributions

All of the implemented code for all components can be found at the GitHub repository: https://github.com/thewyveo/NLP

Please clone repository from GitHub to run the code.

Division of work:

- **Kayra Ö**[redacted] [redacted]: *Topic Modeling Component (code & analysis/report), help with NERC Component (code), setting up GitHub page/integrating pipeline & modules, finalizing poster. 4 datasets, 11 references.*
- M. **Fatih A**[redacted] [redacted]: *Sentiment Analysis Component (code & analysis/ report),Creating the project poster. 2 datasets, 5 references.*
- **Ahmad M**[redacted] [redacted]: *NERC coding and analysis. Helping with the report and poster.*
- **Rais S. F. S**[redacted] [redacted]: Help & *analysis with NERC component, finalizing analysis and report, helping with the poster.*

REFERENCES

[1]Honnibal, M., Montani, I., Van Landeghem, S. & Boyd, A. (2020). spaCy: Industrial-strength Natural Language Processing in Python. , . doi: 10.5281/zenodo.1212303

[2]Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2018, October 11). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv.org. https://arxiv.org/abs/1810.04805

[3]Y. Mejova, "Sentiment analysis: An overview," Comprehensive Exam Paper, Dept. of Computer Science, Univ. of Iowa, Iowa City, IA, USA, Nov. 2009. [Online]. Available: https://www.academia.edu/291678/Sentiment_Analysis_An_Overview

[4] C. J. Hutto and E. Gilbert, "VADER: A parsimonious rule-based model for sentiment analysis of social media text," in Proc. 8th Int. AAAI Conf. Weblogs Soc. Media (ICWSM), Ann Arbor, MI, USA, May 2014, pp. 216-225.

[5]F. Pedregosa et al., "Scikit-learn: Machine Learning in Python Gaël Varoquaux Bertrand Thirion Vincent Dubourg Alexandre Passos PEDREGOSA, VAROQUAUX, GRAMFORT ET AL. Matthieu Perrot Edouard Duchesnay," Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011, Available: https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf?source=post_page

[6]Y. Hou et al., "Bridging language and items for retrieval and recommendation," arXiv preprint arXiv:2403.03952, 2024.

[7] B. Pang and L. Lee, "A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts," Proc. of the ACL, 2004. [Online]. Available: https://www.cs.cornell.edu/people/pabo/movie-review-data/

[8] M. Bakhet, "Amazon Books Reviews," Kaggle, [Online]. Available: https://www.kaggle.com/datasets/mohamedbakhet/amazon-books-reviews

[9] K. Lan, "GOAL: Towards Benchmarking Few-Shot Sports Game Summarization," GitHub, [Online]. Available: https://github.com/krystalan/goal

[10] R. Kulkarni, "Irish Times - Waxy-Wany News," Kaggle, [Online]. Available: https://www.kaggle.com/datasets/therohk/ireland-historical-news

[11] S. Prusty, S. Patnaik, and S. K. Dash, "SKCV: Stratified K-fold cross-validation on ML classifiers for predicting cervical cancer," Frontiers in Nanotechnology, vol. 4, Aug. 2022. [Online]. Available: https://www.frontiersin.org/journals/nanotechnology/articles/10.3389/fnano.2022.972421/full

[12] D. Jurafsky and J. H. Martin, Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models, 3rd ed., 2025. [Online]. Available: https://web.stanford.edu/~jurafsky/slp3/

[13] I. Korshunova and H. Xiong, "Discriminative Topic Modeling with Logistic LDA,", Chapter 4 Logistic LDA, arXiv preprint arXiv:1909.01436, Sep. 2019. [Online]. Available: https://arxiv.org/abs/1909.01436

[14] D. M. Blei and J. D. McAuliffe, "Supervised Topic Models," Advances in Neural Information Processing Systems, vol. 20, 2007. [Online]. Available: https://proceedings.neurips.cc/paper/3328-supervised-topic-models.pdf

[15] I. Vayansky and S. A. P. Kumar, "A Review of Topic Modeling Methods," Information Systems, Chapter 4.1.1, Latent Dirichlet allocation, vol. 94, 2020, Art. no. 101582.

[16] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet Allocation," Journal of Machine Learning Research, vol. 3, pp. 993-1022, Jan. 2003. [Online]. Available: https://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf

[17] E. F. Dinsa, M. Das, and T. U. Abebe, "A topic modeling approach for analyzing and categorizing electronic healthcare documents in Afaan Oromo without label information," pp. 2-3, Background in Topic Modeling Approaches, Scientific Reports, vol. 14, no. 1, 2024. [Online]. Available: https://doi.org/10.1038/s41598-024-83743-3

[18] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by Latent Semantic Analysis," Journal of the American Society for Information Science, vol. 41, no. 6, pp. 391-407, 1990. [Online]. Available: http://wordvec.colorado.edu/papers/Deerwester_1990.pdf

[19] TU Delft, "Singular Value Decomposition," Interactive Textbooks: Linear Algebra, [Online]. Available: https://interactivetextbooks.tudelft.nl/linear-algebra/Chapter8/SingularValueDecomp.html

[20] X. Zhang and E. Milios, "MPTopic: Improving topic modeling via Masked Permuted pre-training," pp. 2-3, arXiv preprint arXiv:2309.01015, Sep. 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2309.01015

[21] D. M. Blei and J. D. McAuliffe, "Supervised Topic Models," Advances in Neural Information Processing Systems, vol. 20, 2007. [Online]. Available: https://www.cs.columbia.edu/~blei/papers/BleiMcAuliffe2007.pdf