NLP Models for Classifying Duke University Student Group Events

Lisa Wang, Gavin Li, Xiyue (Vivian) Zhang

Introduction

Our project is dedicated to gathering and analyzing event information from Duke University websites with the goal of developing robust models for classifying events based on their title and descriptions. The meaningfulness of this project lies in its potential to contribute to more effective event management and organization. As the volume of event data continues to grow, automating the categorization process becomes increasingly essential. Accurate event classification not only streamlines information retrieval for users but also facilitates targeted event promotion and resource allocation.

This undertaking holds significant value in the realm of natural language processing and event categorization. By leveraging generative probabilistic modeling through Latent Dirichlet Allocation (LDA) coupled with logistic regression, and employing a discriminative neural network approach with a transformer model, we aim to enhance our understanding of diverse event types and improve the accuracy of classification systems.

Data Preparation

Data Source

- Duke Student Group: https://dukegroups.com/events
- Description: Events posted by Duke student groups. There are several sources of data for Duke Events. But the event at Duke Student Group contains rich and complete information. So we can use the information gathered from this group to train the model and apply it to other groups.
- Timeline: The earliest event data is from October 2019, with a total of 19,065 events (as of November 2023).
- Note: Some event locations are private and require login to view, so private location data is not collected.

Data Retrieval

- To collect event information, we employed a web scraping approach using Python Beautiful Soup, Requests, and Selenium library.
- The script navigates through the Duke Groups website, targeting specific event types, and collects event IDs. We collected event IDs for five predefined event types: Health/Wellness, Social, Workshop/Short Course, Lecture/Talk, and Panel/Seminar/Colloquium.
- These IDs are then used to retrieve detailed event information from individual event pages. Information includes event name, details, host, location, date, time, and a link to the event. The retrieved data is stored in CSV files, each row corresponding to a specific event type.

Data Cleaning

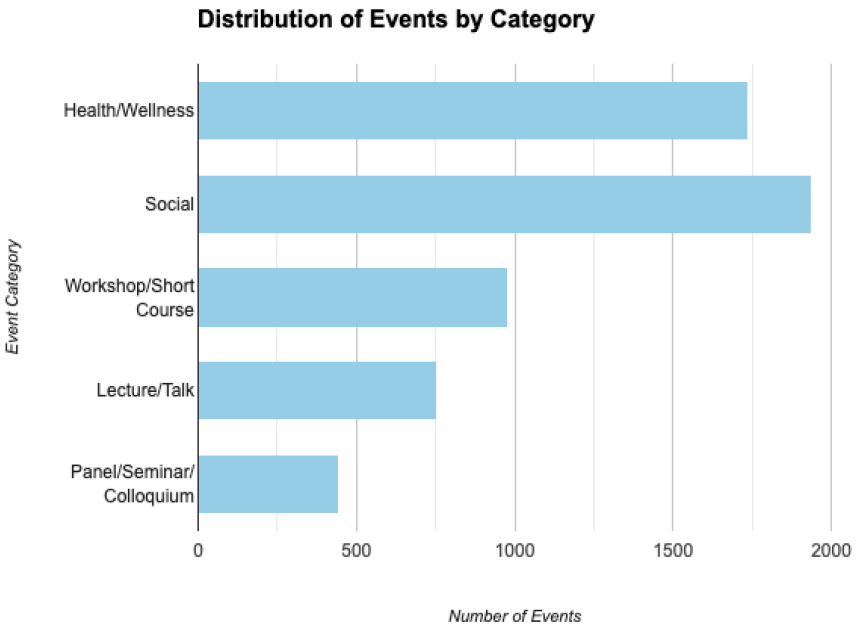
The collected event information may contain noise or inconsistencies. To ensure the quality of the dataset, we performed the following data cleaning steps:

- Removing missing data: Some events might have missing or incomplete information. We checked for such cases and either excluded them or imputed missing values where appropriate. Seven rows with missing event details information were moved.
- Standardizing Event Types: Event types were predefined and hardcoded into integer values during data collection. We ensured consistency in event type labels for further analysis.
- Duplicate Event Check: Our dataset does not contain identical events; each event is associated with a unique type. However, there are instances where events appear repetitively or in series. While we haven't removed these events at present, it is acknowledged that addressing such repetitive occurrences may be considered in the future to enhance model performance.

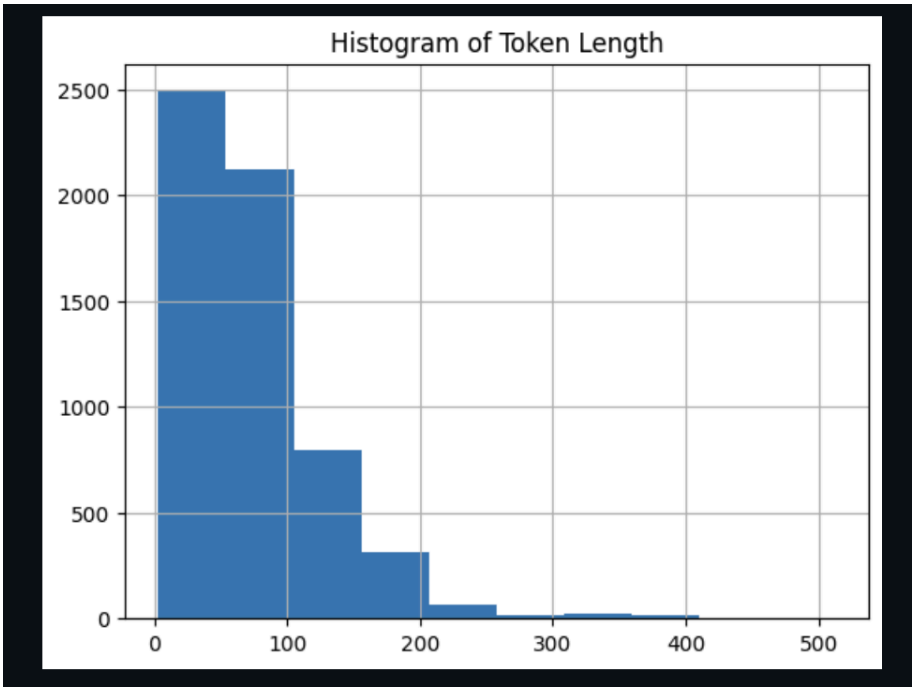
Data Overview

We conducted thorough Exploratory Data Analysis (EDA) to gain a deeper understanding of the dataset, informing more informed decisions during model training and tuning.

The dataset exhibits imbalanced event distribution, with Health/Wellness and Social event types dominating while Workshop/Short Course, Lecture/Talk, and Panel/Seminar/Colloquium categories are underrepresented. Attempts to address this imbalance, such as merging the latter three into an "academic" category, proved less effective than anticipated due to their inherent dissimilarity. For the project's learning goals, the decision was made to exclusively train the model on the more prevalent Health/Wellness and Social event categories, aiming to improve performance within this constrained scope.



Two text columns crucial for predicting event types are the event's title and description. While the title contains more key information, it is shorter and less inclusive. To address this, a decision was made to concatenate the title and description columns, forming a new column for prediction. Post-tokenization, the length distribution of this new column is right-skewed, with a prevalence of shorter events. In consideration of a balance between information retention during training and computational efficiency, a maximum length of 200 was chosen for our BERT model, optimizing the trade-off between training effectiveness and resource utilization.



Methodology

Generative Model: Latent Dirichlet Allocation + Logistic Regression

In our generative model approach, we begin by filtering out entries with missing event descriptions or with missing type identifiers to maintain data integrity. We focus on a subset of the data with type identifiers 1 and 2, narrowing our analysis to two event categories. Once the data is curated, we tokenize the event descriptions into individual words, creating a structured list of documents for further processing. Correspondingly, we adjust the type identifiers to conform with Python's zero-based indexing, which will serve as our target labels for classification.

To facilitate our topic modeling, we construct a term dictionary and a document-term matrix, which are essential components for training our Latent Dirichlet Allocation (LDA) model. The LDA model is tasked with identifying the latent topics within each event description. We specify the model to discern between two principal topics, aligning with the number of event

categories under consideration. The probabilistic topic distributions generated by the LDA model are then harnessed as features for a logistic regression classifier. To match the required input format of the LogisticRegression model class in scikit-learn, we converted the list of topic weights into a dense feature matrix, filling in any gaps with zeros to account for topics that might be absent from certain documents.

Following the preprocessing, we split our dataset into training and testing subsets, applying a conventional 80-20 split. A Logistic Regression classifier is then trained on the training set, fine-tuning its parameters to fit the topic-based features extracted from the event descriptions. Finally, we evaluate the classifier's performance on the test set.

Discriminative Model: Transformer

For the discriminative neural network, we selected BERT(Bidirectional Encoder Representations from Transformers) due to its exceptional contextual understanding of language, pre-trained representations, and task-agnostic nature. BERT's transformer architecture excels in capturing intricate relationships within event descriptions, making it well-suited for accurate and versatile event classification tasks, especially when dealing with varied and nuanced language patterns on Duke websites.

The model used in the code is BertForSequenceClassification from the Hugging Face Transformers library. This model is based on the Transformer architecture, which relies heavily on self-attention mechanisms. In particular, the attention mechanisms, especially self-attention, allow the model to weigh the importance of different words in a sequence when making predictions. BERT uses a multi-head self-attention mechanism to capture contextual relationships between words bidirectionally.

Texts are tokenized using the BERT tokenizer (BertTokenizer.from_pretrained("bert-base-uncased", do_lower_case=True)). It creates PyTorch datasets and data loaders for training and validation. The BERT model is initialized for sequence classification with a specified number of labels (num_labels=3). We used BERT tokenizer to process the training and testing text.An AdamW optimizer and a linear learning rate scheduler are set up.

The chosen loss function for this model is the cross-entropy loss. Cross-entropy loss is chosen for its suitability in classification tasks, penalizing deviations between predicted probabilities and true class distributions, promoting efficient gradient-based optimization, and stabilizing training. In the training loop, the code computes this loss during each iteration, and the optimizer aims to minimize it. The cross-entropy loss is well-suited for classification tasks, penalizing deviations between predicted probabilities and actual class labels. As the model undergoes training, the loss initially tends to decrease, indicating that the model is learning and improving its predictive capabilities.

Opting for 10 epochs, we observed a substantial performance boost in the initial 2-3 epochs, with accuracy climbing from around 87% to approximately 95%. However, the rate of improvement slowed in later epochs, and there was a slight performance dip, potentially indicative of the model attempting to leverage more complex patterns. Considering the diminishing returns in later epochs, we concluded that 10 epochs provide a sufficient training duration to achieve a robust and effective model.

To optimize computing power for model training, we utilized a GPU environment. We ensured the model and associated computations were efficiently moved to the GPU for accelerated processing.

Experiment Setup
In our endeavor to assess the resilience of our classification models to syntactic variations, we introduced a controlled perturbation into our dataset. This was achieved by randomly shuffling the order of words in the *title_detail* field of each entry, which contains the event descriptions. This shuffling process was systematic and applied uniformly across the dataset, ensuring that while the individual words in the titles remained intact, their original sequencing was disrupted. The intent behind this was to simulate scenarios where syntactic structures are compromised, yet the semantic elements required for classification are preserved. The shuffled dataset thus served as a novel test bed to evaluate the robustness of our generative and discriminative models against the loss of syntactic coherence.

Results

	Accuracy	Time to Train	Precision	Recall	F1 Score
LDA + Logistic Regression - Real Data	76%	<1s with CPU only	0.76	0.75	0.75
Transformer - Real Data	97%	350 seconds on a system with 6 cores, 1 GPU, and 128 GB of memory	0.96	0.95	0.94
LDA + Logistic Regression - Synthetic Data	77%	<1s with CPU only	0.77	0.77	0.77
Transformer - Synthetic	95%	350 seconds on a system with 6 cores,	0.94	0.91	0.91

Data		1 GPU, and 128 GB of memory.			

Analysis

We shuffled word order in each input data in both the transformer model and dirichlet model: The **transformer model** has an accuracy of 96.59% for real data and 95.23% for synthetic data, which are both high suggesting that the model has a strong predictive performance. The **dirichlet model** has an accuracy of 76% for real data and 77% for synthetic data, which is lower than the transformer model.

1. **Date, Time, and Computational Requirements:**

- **Transformer Model:** Requires much more computation time. We trained on a GitHub codespace with 6 cores, 1 GPU, and 128 GB of memory, but it still took 6 minutes to run all of the ten epochs. They also tend to be more data-hungry, benefiting from recent and large datasets.
- **Dirichlet Model:** Requires much less computational time and less resources than the transformer model. It can work well with smaller datasets and does not necessarily benefit from the recency of data.

2. **Performance Change Due to Word Shuffling:**

- Word shuffling seems to have a slight impact on the Dirichlet model, with accuracy remaining almost the same (76% to 77%). This could be because LDA focuses on the presence of words rather than their order, making it resilient to this kind of perturbation.
- Transformers are designed to capture sequential information and contextual relationships between words. The attention mechanism allows the model to weigh the importance of each word relative to others in a sentence, so shuffling words disrupts these relationships. We expect a more substantial drop in performance when words are shuffled, but this not too obvious change (97%-95%) might be because the dataset's text is short, so word order does not significantly alter the meaning or the detectability of the topic.

3. **Performance Change on Synthetic Data:**

- To validate the performance of our two models further, we used synthetic data. These are not generated by real-world events, but artificially created by ourselves using different techniques. We explored two methods for synthetic data generation: firstly,

by probabilistically generating new sentences based on the likelihood of each token appearing, and secondly, by incorporating both textual and Latent Dirichlet Allocation (LDA)-derived features. In the latter approach, we applied LDA features, introduced random labels, and iteratively repeated this process to generate multiple synthetic samples.

- Nevertheless, our LDA model achieved only approximately 50% accuracy, while BERT demonstrated around a 70% accuracy rate. The underperformance of the LDA model was unexpected, especially considering the probabilistic model's anticipated superiority on synthetic data generated through its approach. Potential factors contributing to this discrepancy include data noise, suboptimal tokenization, or peculiarities in our training data. We remain committed to investigating and addressing these issues, continuously diagnosing and refining our model for enhanced performance.

4. **Interpretability:**

   The Dirichlet Model is more interpretale since it's based on probabilistic topic distributions. **The Transformer Model** is less interpretable due to the complexity of the model's multiple layers and attention mechanisms.

5. **Real-world Applications Implications:**

   For tasks like event description classification, where the context provided by the order of words can be crucial, transformer models would generally be preferred. However, if computational resources are limited or if interpretability is more important, an LDA-based approach could be more suitable.

6. **Other Pros and Cons:**

- **Transformer Model Pros:** High accuracy, ability to capture complex patterns, and state-of-the-art performance on many NLP tasks. However, it requires significant computational resources, is less interpretable, and may overfit on smaller datasets without proper regularization.
- **Dirichlet Model Pros:** More interpretable, lower computational requirements, and can perform well on smaller datasets. However, it has lower accuracy for classification tasks and might oversimplify complex patterns.

Conclusion

The comparison between the Transformer and LDA models underscores their respective trade-offs. The Transformer excels in capturing intricate patterns and sequential information but necessitates greater computational resources, while the LDA model, though less accurate, offers interpretability and performs well with smaller datasets. Consideration of task requirements, computational resources, and the need for interpretability should guide the choice between these models in real-world applications.

Our research holds significance as it leverages real-world, up-to-date data collected from websites, providing novel insights into an unexplored topic. However, limitations exist, particularly in the data's source—events primarily posted by students, potentially differing from public or academic events.

Future improvements can enhance our model by expanding the dataset size, conducting deeper data analysis and cleaning, and incorporating additional information such as location, host, and event time for prediction. Exploring other NLP tasks, including summarization and text generation, and experimenting with alternative models can further refine our approach.

In conclusion, our study sheds light on the strengths and limitations of generative and discriminative models in event classification, empowering practitioners to make informed decisions in natural language processing tasks.

Appendix

GitHub Repo: https://github.com/nogibjj/Event_classifier/tree/transformer

Model performance Screenshot:

LDA on real data

	precision	recall	f1-score	support
0	0.78	0.66	0.72	344
1	0.74	0.84	0.78	389
accuracy			0.76	733
macro avg	0.76	0.75	0.75	733
weighted avg	0.76	0.76	0.75	733

LDA on shuffled data

	precision	recall	f1-score	support
0	0.78	0.71	0.74	344
1	0.76	0.82	0.79	389
accuracy			0.77	733
macro avg	0.77	0.77	0.77	733
weighted avg	0.77	0.77	0.77	733

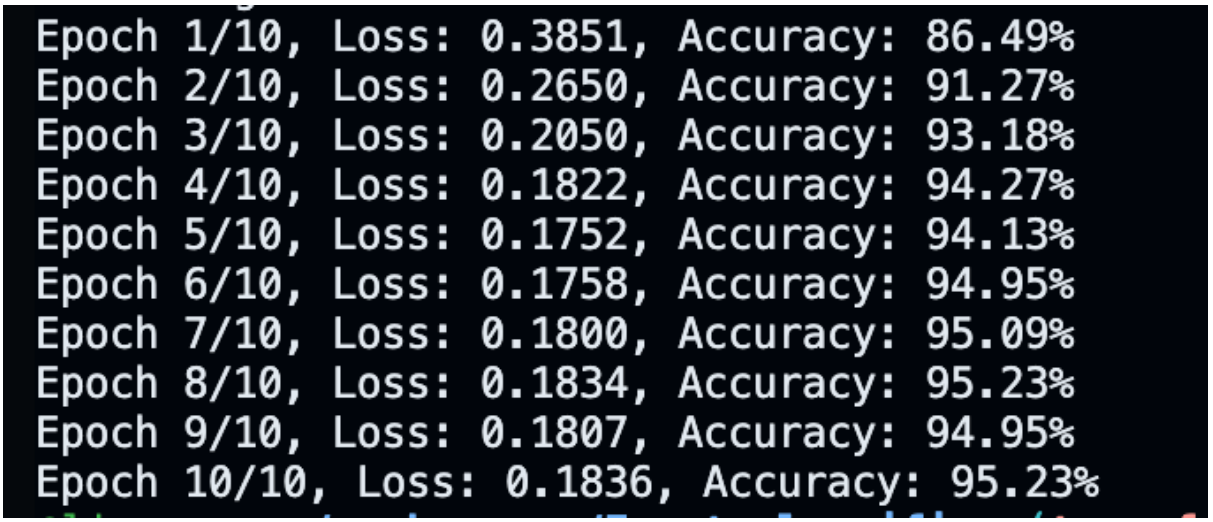
LDA on synthetic data

	precision	recall	f1-score	support
0	0.50	0.37	0.43	2910
1	0.50	0.63	0.56	2929
accuracy			0.50	5839
macro avg	0.50	0.50	0.49	5839
weighted avg	0.50	0.50	0.49	5839

Transformer on real data

```
Epoch 1/10, Loss: 0.3099, Accuracy: 91.54%
Epoch 2/10, Loss: 0.1749, Accuracy: 94.68%
Epoch 3/10, Loss: 0.1398, Accuracy: 95.09%
Epoch 4/10, Loss: 0.1241, Accuracy: 95.77%
Epoch 5/10, Loss: 0.1166, Accuracy: 96.04%
Epoch 6/10, Loss: 0.1175, Accuracy: 96.04%
Epoch 7/10, Loss: 0.1135, Accuracy: 96.59%
Epoch 8/10, Loss: 0.1108, Accuracy: 96.45%
Epoch 9/10, Loss: 0.1111, Accuracy: 96.59%
Epoch 10/10, Loss: 0.1110, Accuracy: 96.59%
```

Transformer on shuffled data:



Transformer for synthetic data: