LLMs vs Established Text Augmentation Techniques for Classification

Andrew Lin Kevin Oliveira Downing Thomas Ji

1. Introduction

In many real-world applications of machine learning, collecting a sufficiently large and diverse dataset is often a primary challenge. This issue is particularly pronounced in text classification tasks, where data can be sparse, imbalanced, and insufficient for effective model training. These limitations often lead to overfitting, restricted generalization, and poor performance on underrepresented classes. Recent advancements in natural language processing (NLP) have expanded the potential of text augmentation—techniques that synthetically expand datasets—to address these challenges effectively.

Classical text augmentation techniques, such as random word swaps, insertions, and MixUp augmentation, have demonstrated benefits in mitigating data limitations and improving robustness in text classification tasks [8]. More sophisticated strategies, such as backtranslation and paraphrasing, have shown promising results in enhancing model performance by diversifying the training data [1, 8]. Backtranslation, for instance, has been successfully used in domains like hate speech detection and sentiment analysis, where data augmentation improves model robustness and accuracy [1, 7]. Similarly, studies have highlighted the effectiveness of data augmentation for improving neural machine translation systems [5].

In parallel, the rise of large language models (LLMs) has introduced new possibilities for generating high-quality, task-specific text. These models provide a level of linguistic sophistication and contextual awareness that was previously unattainable. Recent work compares the performance of LLM-based text augmentation with non-LLM methods, showing that while LLMs can generate diverse and high-quality data, their computational costs must be weighed against their benefits [2].

This paper investigates the effectiveness of non-LLM versus LLM-based text augmentation methods in improving classification performance on two distinct datasets. The first dataset, a Spotify song dataset, includes lyrics for genre classification—a task with practical applications such as personalized recommendations and enhanced user experience on music platforms. The second dataset, containing job descriptions from LinkedIn, is used to classify job titles, supporting job search and categorization processes on career platforms.

To assess the impact of different augmentation techniques, we will train and evaluate two models: a linear model (SVM) as a baseline and a fine-tuned BERT model, which represents a robust standard in NLP. We will evaluate each model using multiple augmentation methods, both non-LLM and neural, comparing their performance in terms of F1 score and accuracy. By comparing the strengths and limitations of classical and LLM-based augmentation techniques, this study aims to provide a comprehensive analysis of their effectiveness for diverse text classification tasks.

2. Related Works

Data augmentation was initially proposed to improve image datasets; however, many recent studies demonstrate the performance benefits of data augmentation on textual datasets for NLP. It has also been shown that improvements are achievable with simple, low-cost augmentation techniques. A 2019 paper by Jason Wei and Kai Zou presents several easy data augmentation techniques [8]. These techniques include synonym replacement, random insertion, random swap, and random deletion. Each works by randomly performing said operations on words within a sentence or text data point. For example, random swap randomly swaps two words in a sentence, synonym replacement replaces words with synonyms, etc. Wei and Zou tested an RNN and CNN across five text classification datasets both with and without EDA. The datasets included sentiment analysis, customer reviews, subjectivity/objectivity, question type, and pro-con. Results showed consistently higher performance with EDA, with an average accuracy improvement of 3% on 500 data point-sized subsets of the original datasets. One caveat, however, is that only a 0.8% average accuracy improvement was reported on the full datasets, suggesting these techniques are more useful on smaller datasets. Furthermore, these tests were done on untrained as opposed to pre-trained models. It is possible that pre-trained models do not benefit from data augmentation in the same way as untrained models.

Neural text augmentation has been important to give better classification. When done correctly, neural methods are

able to generate semantically similar sentences whilst diversifying sentence structure and vocabulary. One key concern about neural augmentation models is their latent costs (e.g. time, money, hardware). Naturally, there resides a trade-off between the efficiency of the newly generated text when compared to the faster and cheaper classical methodologies [2].

One common neural text augmentation method is with backtranslation. The high level idea is to translate a source text into a language, then back to its source language. One key advantage of this method is the low cost of modern implementation of machine translation models, such as Google Translate. Additionally, backtranslation has the capability to be able to generate a multitude of sentences from a single seed sentence by translating to multiple languages [3]. Backtranslation has been shown to improve training performance in many different domains over classical augmentation methods [1, 7]. Backtranslation is often combined with other augmentation techniques, such as classical methods (e.g. word swap) [7] or with a paraphrasing model [1] to maximize the number of resulting data points. One limitation of this approach is that backtranslated sentences can lack diversity, with returned sentences being too similar to their original counterparts [5].

Another promising approach to neural text augmentation is the use of large language models (LLMs) for paraphrasing. Generative LLMs create diverse text samples by rephrasing initial sentences, having the potential to enhance the robustness of classifiers trained on the augmented data. However, the research validating a clear cost-benefit advantage of LLM-based augmentation over established methods, such as backtranslation, remains limited. Recent studies on LLM-based augmentation indicate that its benefits could be dependent on the context as, for instance, effective on a very small number of seed samples (5-20 labels) but less prominent for larger label sizes [2]. However, when in comparison to backtranslation and classical augmentation methods, LLM-based techniques yielded similar or lower accuracy models suggesting that established methods remain competitive, particularly given their generally lower computational cost.

3. Datasets

3.1. LinkedIn Dataset

The LinkedIn dataset was prepared for text classification by removing duplicate job postings to eliminate redundancy and bias. The dataset was then filtered to retain the top 10 job titles ranked by popularity, with each title capped at 1,000 postings and descriptions standardized to a length of 3,000 to 5,000 characters for consistency. There were a lot of job titles with duplicated descriptions which were removed. Titles were grouped by other titles with basic synonyms, abbreviations, or tenses; for example, nursing and nurse or human resources and hr. This was done for 10 selected titles of occupations that were quite distinguishable from each other. The resulting dataset had a small class imbalance with only three classes being under the cutoff of 1,000 postings with the most underrepresented class still having 441. This was done to see how text augmentation works for datasets with low class imbalance and whether LLMs could manage to improve accuracy for which typical text augmentation methods cease to do for low imbalances.

The resulting 10 job titles are as follows (italics imply underrepresentation):

- Sales
- Nurse
- Human Resources
- Accountant
- Developer
- Project Manager
- Therapist
- *Customer Service*
- *Attorney*
- *Teacher*

3.2. Spotify Dataset

The Spotify dataset merges two publicly available datasets from Kaggle: one containing over 70,000 Spotify songs labeled with genres and another with over a million songs including lyric data. Both datasets were created using the Spotify API and subsequently merged based on song title and album, resulting in a dataset of approximately 50,000 songs with both lyric and genre information across over 120 genres.

Non-English songs and genres were excluded to minimize the confounding effects of language on genre classification, particularly for genres like Brazilian, J-Pop, and Latin music. Non-English songs were identified using the Google Translate API, alongside a heuristic that classified songs with more than 5% non-ASCII characters as non-English. Additionally, genres with significant overlap or ambiguity (e.g., EDM, electronic, electro) were consolidated into broader classifications. After these steps, the dataset was reduced to around 25,000 usable songs spanning 23 genres.

The dataset exhibits class imbalance, with the largest genre containing approximately 4,400 songs, while most genres have fewer than 1,000. To streamline hardware and time requirements for model training, we selected the five most frequent genres (1,600–4,500 songs per genre) and the five least frequent genres with at least 100 songs (100–400 songs per genre). A minimum threshold of 100 songs per genre was maintained to ensure sufficient diversity across genres and artists.

The resulting ten genres are as follows (italics imply un-

derrepresentation):

- Electric
- Heavy Metal
- Rock
- Traditional
- Pop
- *R&B*
- *Comedy*
- *Funk*
- *Ambient*
- *Jazz*

See the appendix for how the larger classifications groups genres.

4. Technical Approach

4.1. Classical Algorithm

For the basic augmentation technique, random insertion, deletion, synonym replacement, and random swapping will be performed on words in each string (entire song lyrics for the Spotify dataset and job descriptions for LinkedIn) to generate new data points. The number of augmentations per string is proportional to the word count, ensuring consistent noise per data point. For example, given a word count of n words we introduce $n/10$ random insertions, $n/10$ random deletions, $n/10$ random swaps, and $n/10$ synonym replacements. To implement each of these techniques, the TextAugment library was used. This library includes implementations from the paper from Marivate and Sefara [6]. Specifically, its implementation of the aforementioned EDA techniques was used here. The basic algorithm was as follows.

```
classically_upsample(string):
    num_augs = int(0.1*num_words)

    new_s = string
    new_s = random_insertion(new_s, num_augs)
    new_s = random_deletion(new_s, 0.1) #
        probability of randomly deleting each
        word
    new_s = random_swap(new_s, num_augs)
    new_s = synonym_replacement(new_s, num_augs)

    return new_s
```

[1]

[1]Note about mixup: The original plan was to work with an existing mixup implementation based on the paper from Guo et. al [4]. Due to the fact that mixup works with the text embeddings themselves as opposed to the original input strings, mixup would need to be dynamic at training-time. This drastically increases runtimes, making hyperparameter tuning much more difficult. Additionally, this setup made it difficult to augment specific classes in the presence of random batch learning. In other words, smaller classes couldn't be upsampled more than larger classes. Although the setup was working, only suboptimal performance was achieved and after many hours, the decision was made to stick to the classically augmented set as the baseline text augmentation method.

4.2. Backtranslation Algorithm

For neural augmentation, the baseline method will be backtranslation via the Google Translate API, selected for its low cost and multi-language support. Our code will randomly choose languages for translation before translating back to English. For LLM-based augmentation, we used Gemini 1.5 Flash to generate additional descriptions that replicate the tone and style of existing entries. We chose Gemini Flash for its speed and its low cost.

```
ASCII_Ratio(sentence)
        total_asciis = 0
        for character in sentence
                if character is not ascii
                        total_asciis += 1
        return total_asciis / sentence.length

BacktranslateSentence(sentence) {
        for language available in Google
            Translate's API {

        translated_sentence = translate(sentence,
             language)
        backtranslated_sentence = translate(
            translated_sentence, ``english'')
        EPSILON = 0.01
        if ASCII_Ratio(backtranslated_sentence) >
             ASCII_Ratio(translated_sentence) -
            EPSILON {
                return backtranslated_sentence
        }
    }
}
```

4.3. LLM Algorithm

For augmentation with LLMs, the Gemini API was used due to its free tier option which made doing large batches of data free although significantly slower due to quota limits. Since, only small changes were desired in the dataset so as to not lead to disruptive misclassification, the following prompt was used: "Rewrite the following [type of data (job postings or song lyrics)] by making small changes: [data]." A standard temperature of 1 was used with top_p=0.95 and top_k=40; too much randomness could have led to augmented datasets differing too much while too little would have just duplicated the result such that the default parameters of Gemini were used.

When augmenting song lyrics, certain songs contained language that was deemed inappropriate by the Gemini API. In these circumstances, that song wouldn't be augmented, requiring other songs that don't contain such language to cover the gap and pass through Gemini. This would in turn create a bias for songs that don't contain "inappropriate" language. There were also issues with timeouts and approaching quota limits which required proper error handling to deal with these issues.

Even under the free plan, Gemini could only handle 15 requests per minute and a total of 1,500 requests per day.

Paid options would provide a much larger bandwidth with 2,000 requests per minute with $0.075 / 1 million tokens for inputting pricing and $0.30 / 1 million tokens for output pricing leading to huge batches of augmented data being more expensive compared to the other approaches.

4.4. Combination of Algorithms

To augment the data, a similar procedure was followed for each technique. Given a dataset, a single data point can be used to "upsample" or create new data points. This will be done multiple times for multiple data points within each class for whichever classes have below a certain threshold of data points that we require. For both datasets, we aim to have at least 1000 data points per class.

After augmentation, a pre-trained uncased BERT classifier from Hugging Face will be fine-tuned via cross-entropy loss on LinkedIn and Spotify datasets, both unaugmented and augmented. We will compare training, validation, and testing accuracies, as well as F1 scores. Initial training will run for 3 epochs, with up to 10 evaluated using early stopping.

Additionally, a linear model will be used to further explore the differences between augmentation methods. We will implement this model in PyTorch as a one-layer (no hidden layer) linear network utilizing multi-class hinge loss. Inputs to this model will be Tf-Idf encoded using sklearn's TfidfVectorizer. For both models, we will use the Adam optimizer.

All training was done on a machine with an RTX 3070 with 16GB of VRAM and 5888 CUDA Cores, an i7-10700 with 8 cores and a clock speed of 3.8GHz, and 16GB of RAM.

5. Results

After hyperparameter tuning, the learning rate 5e-5 with batch sizes of 8 was found to work best across the board for the BERT model. Larger batch sizes proved too costly in terms of memory. For the linear model, the learning rate 1e-2 and batch sizes of 1024 worked best. Epochs took an average of 9 minutes in the unaugmented case for Spotify, but when augmented took an average of 15 minutes. Finally, epochs were 11 minutes for the unaugmented LinkedIn set and 17 for the augmented sets.

See Tables 1 and 2 for an overview of model performance.

6. Discussion

Overall, the training runs that used textual augmentations outperform or match those without augmentation.

Within the Spotify dataset, there was no noticeable improvement in accuracy (all methods around 64%). However, the F1 scores (both Macro and Weighted) improved with both models. The gain in Macro F1 score varied between model, with BERT gaining up to 0.58 (from 0.45) and the linear model gaining up to 0.56 (from 0.53) (see Table 2). This has two possible implications: BERT is able to benefit more from these augmentation methods than the linear model,and the augmentation methods are more effective at improving the model's ability to correctly identify less frequent classes.

Qualitatively, we can see a difference between how BERT is better able to classify underrepresented classes in Figure 5 vs Figures 7 and 8.. Two largely misclassified classes were R&B (0.00 F1 score) and Jazz (0.06 F1 score) with the use of BERT. After using the different training sets, R&B reached above 0.50 F1 score, and Jazz around 0.20. We can view that the model is better understanding these two genres by observing that their respective clusters are more defined.

The SVM model without augmentation demonstrated more uniform F1-scores across its underrepresented classes. This observation aligns with the findings of [2], suggesting that textual augmentation may serve as a form of regularization, enabling BERT to capitalize on its higher performance potential while maintaining more balanced classification across different classes.

Compared to the Spotify dataset, the class imbalance in the LinkedIn dataset was significantly lower. The LinkedIn dataset lowest representative class still had 441 data entries while for Spotify it was 79 with other classes being close to that representation. Hence, with the lower imbalance, augmentation was as necessary as it was in the LinkedIn dataset hence didn't make a significant difference in performance regardless of the technique used. Regardless of the method used SVM or BERT, text augmentation made negligible difference in improving performance. However, the BERT approach did yield better results for text augmentation by about 2-3% implying the method used was more critical for performance than augmentation in low imbalance datasets.

BERT	**LinkedIn**			**Spotify**		
	Accuracy	Macro F1	Weighted F1	Accuracy	Macro F1	Weighted F1
NA	0.96	**0.96**	0.96	0.64	0.45	0.63
Classical	**0.97**	0.96	0.96	**0.65**	0.54	0.63
BT	0.96	0.95	0.96	0.64	**0.58**	0.64
LLM	0.96	**0.96**	0.96	**0.65**	0.57	**0.65**

Table 1. Performance metrics for LinkedIn and Spotify datasets using BERT.

SVM	**LinkedIn**			**Spotify**		
	Accuracy	Macro F1	Weighted F1	Accuracy	Macro F1	Weighted F1
NA	**0.94**	**0.93**	**0.94**	0.60	0.53	**0.60**
Classical	0.93	**0.93**	0.93	0.60	0.54	**0.60**
BT	0.93	0.92	0.93	**0.61**	0.54	**0.60**
LLM	**0.94**	**0.93**	**0.94**	0.60	**0.56**	**0.60**

Table 2. Performance metrics for LinkedIn and Spotify datasets using SVM.

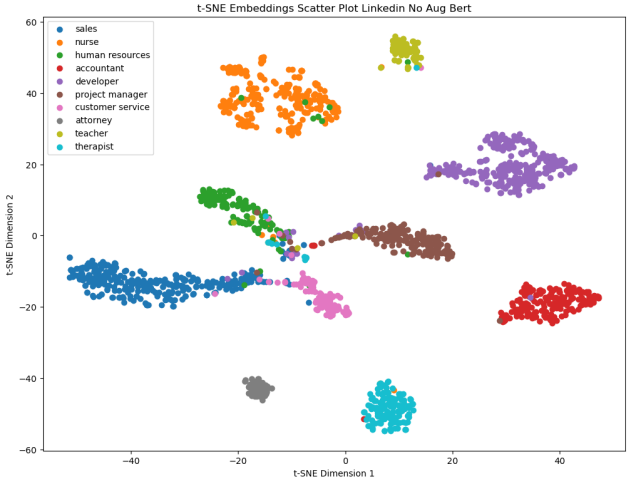


Figure 1. TSNE Embedding of Linkedin without Augmentation

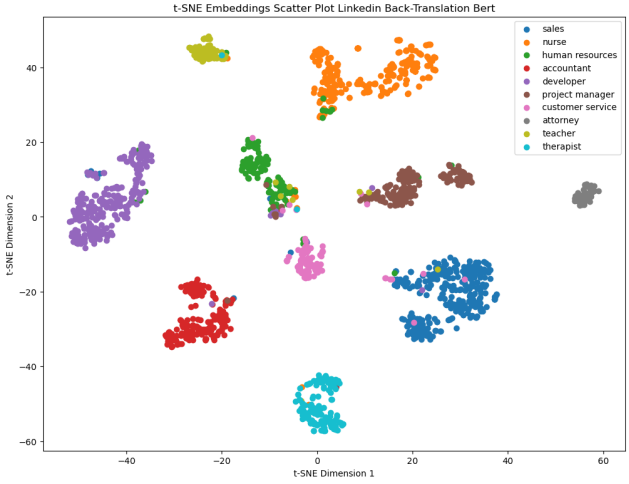


Figure 2. TSNE Embedding of Linkedin with Backtranslation

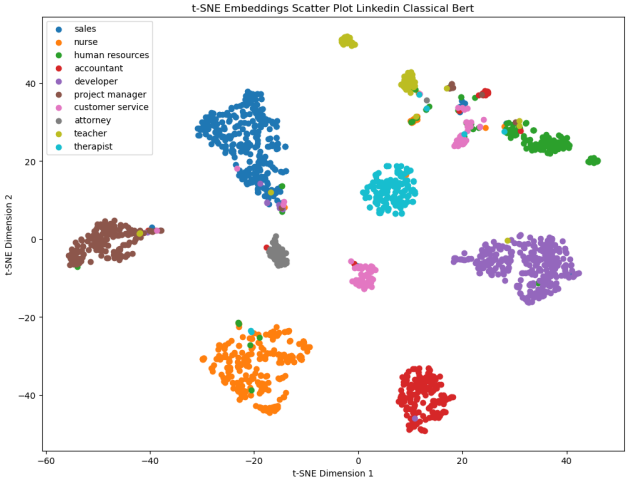


Figure 3. TSNE Embedding of Linkedin with Classical Augmentation

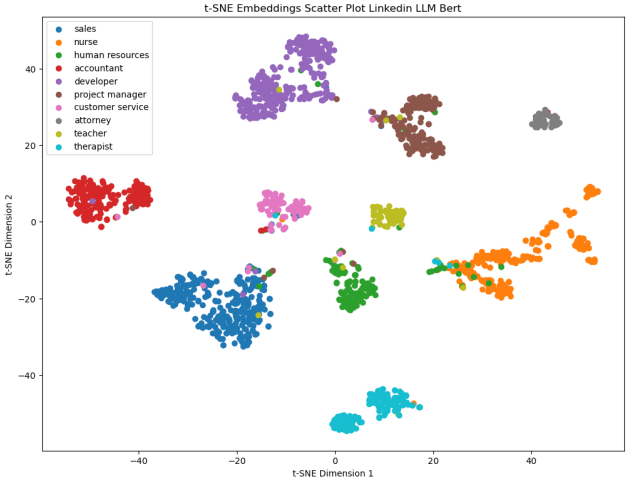


Figure 4. TSNE Embedding of Linkedin with LLM

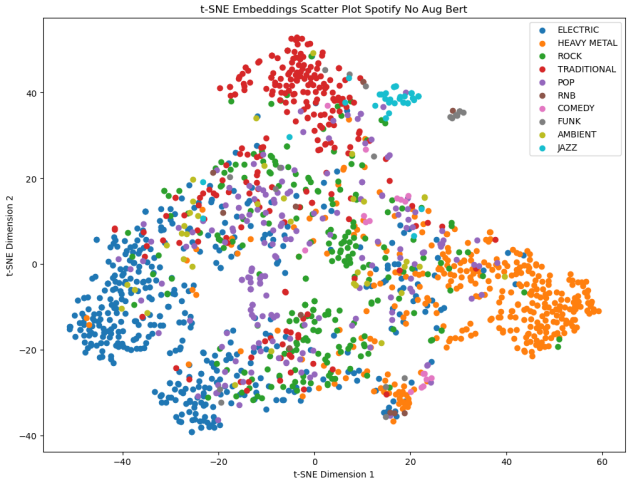


Figure 5. TSNE Embedding of Spotify without Augmentation

Figure 6. TSNE Embedding of Spotfiy with Backtranslation

Although LLMs didn't lead to huge performance, it did match more closely with the results of the non-augmented dataset (see Table 1) above while the other methods appeared to only introduce noise, decreasing accuracy and performance relative to non-augmented. This could be due to the powerful and flexible nature of LLMs to make more sophisticated changes to the dataset as opposed to back translation and classical techniques. The T-SNEs shown in Figures 1-4 however do show generally better separation as there are all distinguish clumps while in the non-augmented, the over represented classes had some ends climbing together such as with sales, hr, project manager, and customer service.

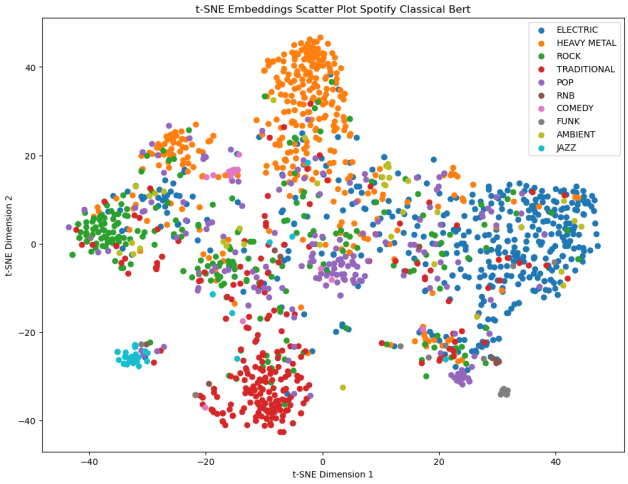


Figure 7. TSNE Embedding of Spotfiy with Classical Augmentation

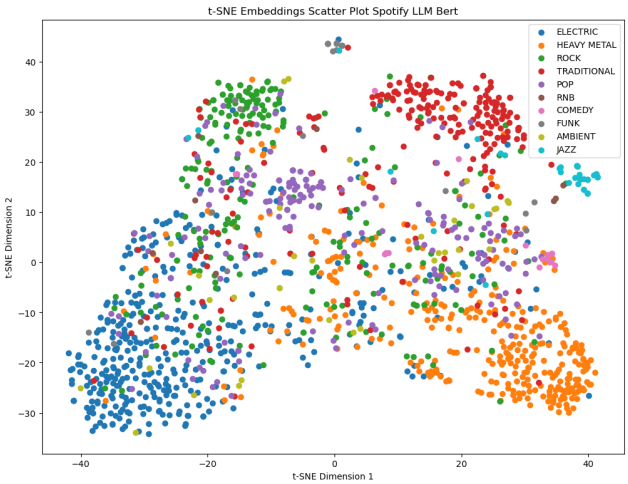


Figure 8. TSNE Embedding of Spotfiy with LLM

7. Conclusion

Overall, we conclude that while small, neural network textual augmentation methods outperform or match classical/no augmentation training settings. In the case of both datasets, we do not see a significant improvement in the use of LLM generated text as opposed to the backtranslation approach. This is corroborated by [2], which found that in the case with more than 20 original texts in the dataset, other augmentation methods were similar in accuracy and holistically cheaper.

The use of LLMs in text augmentation tasks continues to pose several challenges. Two key factors that we experienced in this paper were time and cost factors. With LLMs matching or marginally improving certain metrics, the time it takes to make these API calls is rather significant, par-

ticularly if larger datasets are needed. Smaller, more local models (e.g. translation, paraphrasing) may be more time and cost efficient to accomplish the same goal.

Additionally, content filtering from certain APIs may be problematic for a given classification task. Many LLM APIs such as Gemini and GPT have content filtering to deny prompts they deem inappropriate. This can be problematic when attempting to augmenting data that is known to include inappropriate content as those under represented class won't be able to be properly augmented. In this paper, some songs within the Spotify dataset contained swears and sexual language which was deemed inappropriate for the Gemini API to rewrite. A bias would be introduced to songs that don't contain "inappropriate" language which could cause misclassification.

Given additional time, we would further explore a wider variety of other augmentation methods. This would be able to give us a broader perspective of the LLM's performance in these classification tasks. Furthermore, we would continue to tweak the LLM configurations, including the model itself (e.g. GPT, Claude), prompts, and fine-tuning. Lastly, we would also experiment with different BERT variations to further increase the accuracy on a given task.

8. References

References

[1] Djamila Romaissa Beddiar, Md Saroar Jahan, and Mourad Oussalah. Data expansion using back translation and paraphrasing for hate speech detection. *Online Social Networks and Media*, 24:100153, 2021. 1, 2

[2] Jan Cegin, Jakub Simko, and Peter Brusilovsky. Llms vs established text augmentation techniques for classification: When do the benefits outweight the costs?, 2024. 1, 2, 4, 6

[3] Jean-Philippe Corbeil and Hadi Abdi Ghadivel. BET: A backtranslation approach for easy data augmentation in transformer-based paraphrase identification context. *CoRR*, abs/2009.12452, 2020. 2

[4] Hongyu Guo, Yongyi Mao, and Richong Zhang. Augmenting data with mixup for sentence classification: An empirical study, 2019. 3

[5] Zhenhao Li and Lucia Specia. Improving neural machine translation robustness via data augmentation: Beyond back-translation. In Wei Xu, Alan Ritter, Tim Baldwin, and Afshin Rahimi, editors, *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 328–336, Hong Kong, China, November 2019. Association for Computational Linguistics. 1, 2

[6] Vukosi Marivate and Tshephisho Sefara. *Improving Short Text Classification Through Global Augmentation Methods*, page 385–399. Springer International Publishing, 2020. 3

[7] Alireza Taheri, Azadeh Zamanifar, and Amirfarhad Farhadi. Enhancing aspect-based sentiment analysis using data augmentation based on back-translation. *International Journal of Data Science and Analytics*, Aug 2024. 1, 2

[8] Jason W. Wei and Kai Zou. EDA: easy data augmentation techniques for boosting performance on text classification tasks. *CoRR*, abs/1901.11196, 2019. 1

9. Appendix

Genres and Subgenres

The bold capital letters indicate the larger group of genres underneath it.

ELECTRIC

- breakbeat
- chicago-house
- deep-house
- detroit-techno
- dubstep
- edm
- electro
- electronic
- hardstyle
- house
- idm
- minimal-techno
- progressive-house
- techno
- trance
- drum-and-bass

ALT

- alternative
- indie

AMBIENT

- ambient
- new-age

HEAVY METAL

- black-metal
- death-metal
- grindcore
- heavy-metal
- hardcore
- metal
- metalcore
- industrial

RNB

- r-n-b

ROCK

- alt-rock
- rock-n-roll
- rock
- rockabilly
- hard-rock
- psych-rock
- punk-rock
- grunge

TRADITIONAL

- bluegrass
- blues
- country
- honky-tonk

PARTY

- party
- dance
- club
- disco

POP

- indie-pop
- pop
- synth-pop
- power-pop
- pop-film

CLASSICAL

- classical

JAZZ

- jazz

KIDS

- children
- kids

SHOW-TUNES

- disney
- show-tunes

GOSPEL

- gospel
- soul

FOLK

- folk
- singer-songwriter

REBEL

- goth
- emo
- punk

HIP-HOP

- hip-hop
- trip-hop

FUNK

- funk

CHILL/FOCUS

- chill
- sleep
- study

COMEDY

- comedy

HAPPY

- happy

SAD

- sad

ACOUSTIC

- acoustic
- piano
- guitar

NOT ACTUAL GENRES

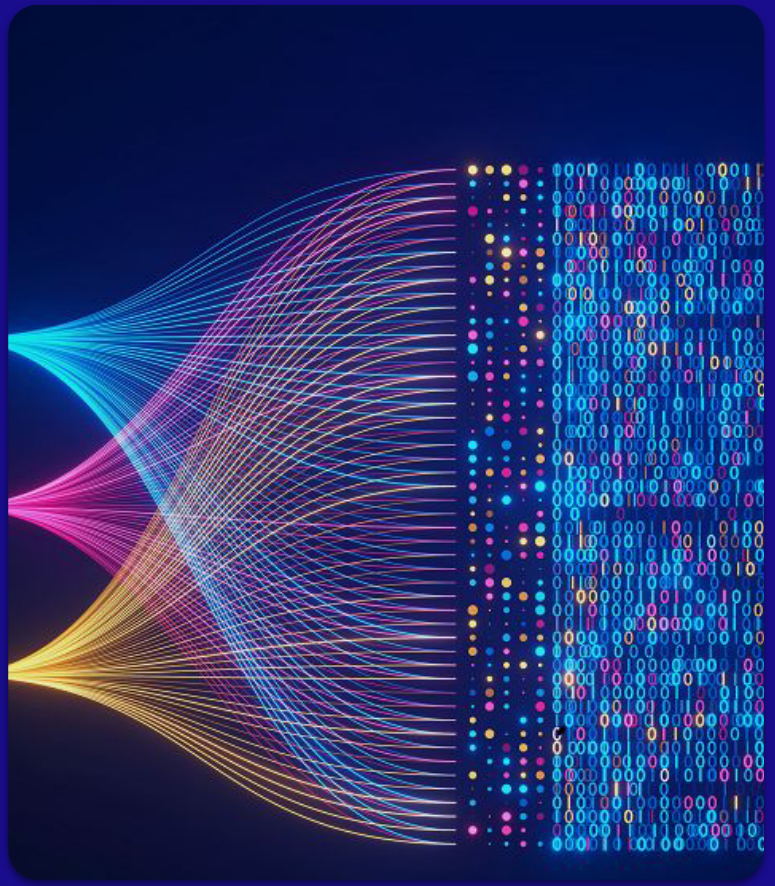
- garage
- groove

CS682

LLMs vs Established Text Augmentation Techniques for Classification (Group 68)

Thomas Ji (tji@umass.edu)
Andrew Lin (andrewlin@umass.edu)
Kevin Oliveira Downing (kjdowning@umass.edu)

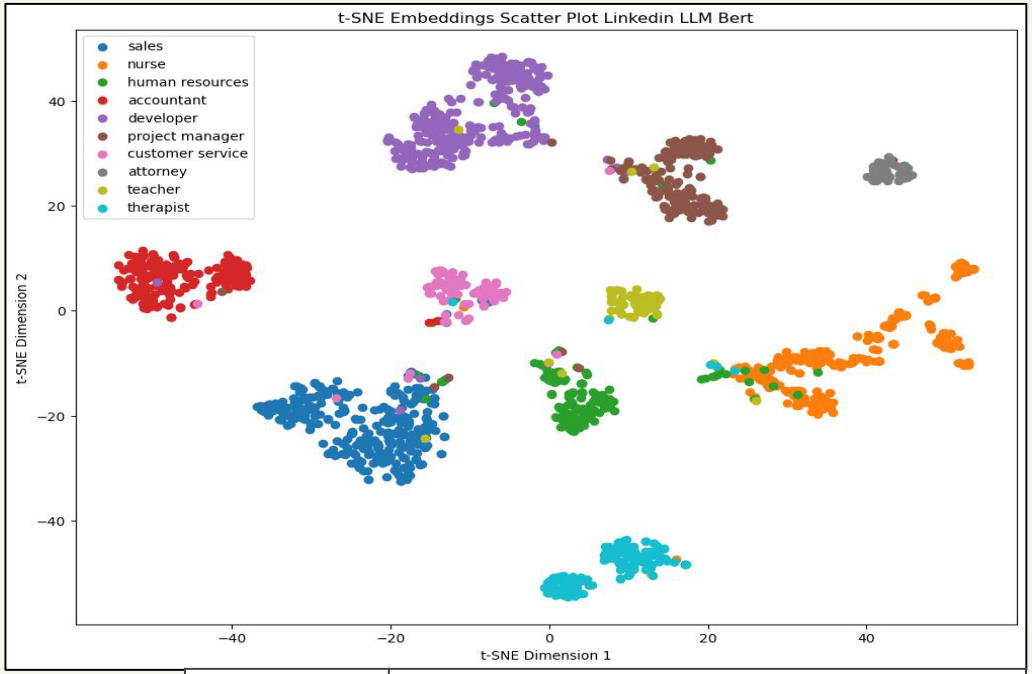
12/03/2025



LinkedIn Results

The LinkedIn dataset was classifying job titles from description.

- Small class imbalance, with many job-related keywords
- Minimal difference between augmentation methods (and no augmentation)
- Clear separation implies that the model is already close to peak performance without augmentation

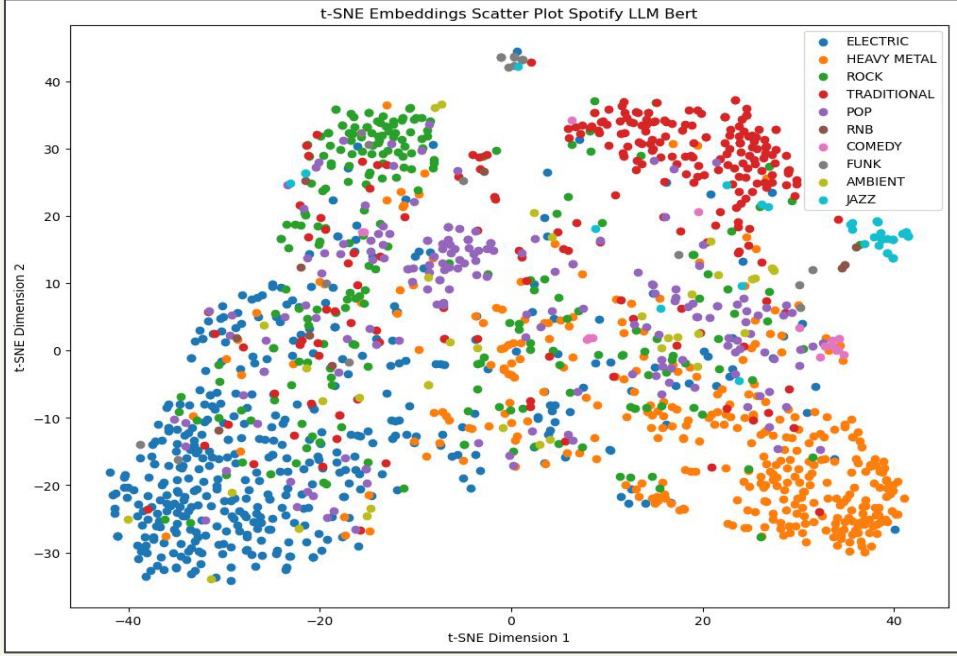


	Linkedin		
	Accuracy	Macro F1	Weighted F1
NA	0.96	**0.96**	**0.96**
Classical	**0.97**	0.96	0.96
BT	0.96	0.95	0.96
LLM	0.96	**0.96**	**0.96**

Spotify Results

The Spotify dataset was about classifying song lyrics with genres.

- Large class imbalance
- LLM and Backtranslation improved F1 scores significantly
- Demonstrates that textual augmentation methods are very relevant



	Spotify		
	Accuracy	Macro F1	Weighted F1
NA	0.64	0.45	0.63
Classical	**0.65**	0.54	0.63
BT	0.64	**0.58**	0.64
LLM	**0.65**	0.57	**0.65**

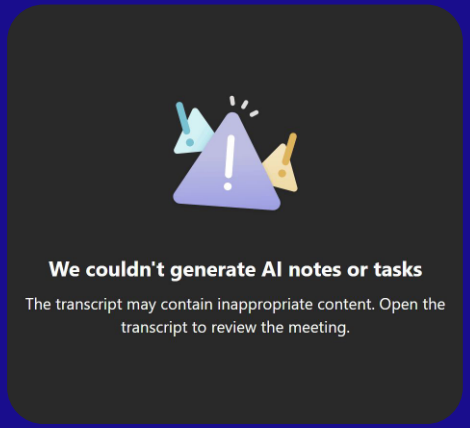
LLM Summary

"Rewrite the following [type of data (job postings or song lyrics)] by making small changes: [data]."

Although LLM performed the best out of the text augmentation methods, the difference is minimal and there are some important drawbacks to consider.

Expensive & Time costly

Most LLMs are reliant on APIs which can prove quite expensive and time consuming over large batches of data.



Inappropriate content filter

Augmentation of datasets that contain inappropriate content can prove problematic as many APIs like Gemini refuse to answer prompts with such language.