LLM - Detect AI Generated Text
Project Report

Zhan Yu
306153151
yuzhan3232@g.ucla.edu

Wenkai Gong
406183261
mga19wg@g.ucla.edu

Zachary Kong
506181247
danielkong@g.ucla.com

Fengzhou Pan
806152050
panfengzhou@ucla.edu

Junyu Ren
006140155
junyuren2000@g.ucla.edu

Abstract

In this project, we address the challenging task of classifying AI-generated text, comparing various approaches that combines fixed word embedding techniques with adaptive classification algorithms. We leveraged Word2Vec and Glove to convert words into vector representations and extends to include context-sensitive embeddings derived from language models such as DistilBERT, BERT, DeBERTa. These embeddings provide a nuanced understanding of language by capturing semantic similarities, serving as a fixed input to our classification framework. In the classification phase, we respectively employed MLP, ML methods(logistic regression, random forest and XGBoost). Also, we tried directly finetune language models. We analyzed distinct advantages of each method, ensuring a robust mechanism for text categorization. Our key findings reveal that combining BERT-like language models with an MLP classifier achieves the best performance, while static methods like word2Vec and Glove also achieve a reasonable result without needing large models. Finetuning language models, despite being time-consuming, significantly enhances performance on complex tasks.

1 Problem Statement

In recent times, the proliferation of large language models has garnered significant public attention. These models are so powerful, and increasingly being used, leading to a surge in machine-generated texts on the internet which are hard to distinguish from those written by humans. The implications of this trend could lead to academic integrity, misinformation, and even security concerns. It is therefore essential to explore effective approaches that can efficiently classify, and flag content produced by these advanced language models.

Our project aims to involve experimenting with various machine learning models, refining their training processes, and performing comparative and algorithmic evaluations. We used two dataset. The first one is from kaggle, which contains students writing essays and AI generated essays. The second one is a larger dataset which was designed by hand. There are 150k human-written and GPT-generated responses to Wikipedia topics.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

2 Literature Review

2.1 Text Representation

2.1.1 Static Methods

Text representation methods play a crucial role in text mining and natural language processing tasks by converting textual data into numerical vectors, facilitating computational analysis and machine learning algorithms. Among the static methods utilized for text representation, word embeddings generated by techniques like word2vec and GloVe have garnered significant attention for their ability to capture semantic relationships between words in a continuous vector space.

Word2Vec, introduced by Mikolov et al. (2013), is a popular word embedding technique that learns distributed representations of words based on their contextual usage in a corpus. The model operates on the principle of predicting the surrounding words given a target word (Skip-gram), or predicting the target word given its context (Continuous Bag of Words, CBOW). By leveraging large text corpora, word2vec efficiently captures semantic similarities and relationships between words, enabling downstream tasks such as text classification and clustering to operate in a semantically meaningful space.

GloVe, proposed by Pennington et al. (2014), is another widely adopted word embedding method that combines global matrix factorization with local context window methods. Unlike word2vec, which relies solely on local context, GloVe incorporates global co-occurrence statistics to derive word embeddings. By representing words as vectors based on the probabilities of their co-occurrence with other words in the corpus, GloVe captures not only syntactic but also semantic relationships between words, yielding embeddings that excel in tasks requiring nuanced understanding of word semantics.

Word2Vec and GloVe lay the foundation by transforming textual data into a mathematical form that captures the inherent meanings and relationships between words. This initial step is crucial for my project as it transitions the unstructured text into a structured format that can be analyzed computationally. The rich semantic information encoded in these embeddings allows my model to grasp the subtleties of language that differentiate human from AI writing styles, such as the diversity of vocabulary and the complexity of sentence structures used by human authors versus the patterns that may emerge in AI-generated texts.

2.1.2 Dynamic Methods

Dynamic text representation methods, specifically BERT and GPT, significantly enhance the analysis of textual data by producing context-dependent embeddings.
BERT (Devlin et al., 2018) revolutionizes context understanding by processing text bidirectionally, allowing for a comprehensive grasp of linguistic nuances. This model's innovative training on masked language modeling and next sentence prediction tasks enables it to capture deep contextual meanings. However, its complexity requires substantial computational resources, posing a limitation.
For our project, aimed at distinguishing between human and AI-generated texts, the contextual sensitivity of BERT embeddings is crucial. We also utilized some other BERT-like methods, such as DeBERTa and DistilBERT. These methods' in-depth contextual analysis aids in discerning subtle linguistic patterns unique to human writing. Their ability to understand and replicate human language nuances is instrumental for accurately classifying text origins, aligning closely with the project's objectives.

2.2 Classification

2.2.1 Traditional Machine Learning Methods

Traditional machine learning classification models have been extensively studied and applied across various domains to solve classification problems. These models, ranging from logistic regressions to support vector machines, form the bedrock of supervised learning. Among them, gradient boosting algorithms stand out due to their enhanced performance compared to other algorithms with simpler

structures.

Logistic Regression is a staple in binary classification tasks, valued for its simplicity, interpretability, and efficiency. It models the probability of a binary outcome as a function of independent variables, offering insights into how predictors influence the outcome (Hosmer Jr, D.W., Lemeshow, S., & Sturdivant, R.X., 2013). Despite its widespread application, it faces limitations in handling nonlinear relationships and high-dimensional data, which can impede its performance in complex tasks like text classification. In the context of our project, which involves text classification, Logistic Regression serves as a baseline model to evaluate the effectiveness of more complex algorithms. Its simplicity and interpretability make it an excellent starting point for understanding the fundamental relationships between textual features and classification outcomes. Moreover, when combined with advanced text representation techniques like word embeddings, Logistic Regression can still offer competitive performance, highlighting its versatility and enduring relevance in machine learning tasks.

Random Forest, developed by Breiman (2001), combines decision trees' simplicity with an ensemble approach, enhancing prediction accuracy and robustness for classification and regression tasks. This method trains multiple decision trees on random data subsets, using averaging or voting for predictions, which mitigates overfitting and boosts performance on unseen data. Random Forest's key strengths include its robustness, ability to assess feature importance, and versatility in handling various data types and tasks. However, its ensemble nature can lead to complexity and reduced interpretability, and it may require significant computational resources. This makes Random Forest particularly suitable for projects needing high accuracy without extensive hyperparameter tuning, and its feature importance evaluation is invaluable for high-dimensional data analysis.

XGBoost, or Extreme Gradient Boosting, is a scalable and efficient implementation of gradient boosting, known for its speed and performance. First introduced by Tianqi Chen (2016), XGBoost follows a traditional tree-based ensemble method to sequentially build a series of decision trees that each correct the errors of its predecessor, while also introduces several key innovations, including regularized learning objectives, tree pruning, and parallelized tree construction, which contribute to its robustness, speed, and accuracy. XGBoost is essential for high-accuracy projects that can manage its complexity and resource demands. Its success across various datasets and tasks makes it a top choice for both competitions and industrial use. While complex, its predictive performance benefits often surpass challenges related to interpretability and tuning. For traditional machine learning, XGBoost enhances the toolkit with cutting-edge algorithms, marking a key advancement in ensemble methods for data scientists.

2.2.2 Deep Learning Methods

Different from traditional machine learning, deep learning uses artificial neural networks with multiple layers to automatically learn feature from the data, which effectively reduce the complexity of feature engineering.

MLP, (Multilayer Perceptron) plays a fundamental role in the field of deep learning, which is one of the earliest forms of deep neural networks. It usually consists of at least three layers of nodes: an input layer, one or more hidden layers, and an output layer. Except for the input nodes, each node is a neuron using a non-linear activation function, such as ReLU, Tanh, and Sigmoid, and utilizes backpropagation for training its layers. MLP performs excellently on classification, regression, and pattern recognition. Although MLPs might face a risk of overfitting training data, we can mitigate the impact through regularization and hyperparameter tuning.

MLP, as the architecture for your classification model, synthesizes the insights gained from Word2Vec, GloVe, and BERT embeddings. It integrates these diverse sources of information, learning from the vast array of features to identify patterns indicative of human or AI authorship. The flexibility of MLPs in handling different input sizes and types, combined with their capacity for modeling complex relationships, makes them ideal for this task. They can efficiently process the nuanced, high-dimensional data provided by the embeddings, facilitating accurate and reliable classification of texts.

3 Methodology

3.1 Dataset

3.1.1 Kaggle Dataset

Size and Composition: The dataset contains over 28,000 essays, split between those written by students (human-authored) and those generated by various LLMs. This size suggests a significant variety of topics, writing styles, and complexities within the essays, making it a rich resource for training and testing detection models. Features: text: Contains the full text of an essay. This feature is crucial as the model's ability to detect the origin (human vs. LLM) will heavily depend on analyzing linguistic patterns, complexity, creativity, and possibly errors or idiosyncrasies in the writing. generated: This binary target label indicates the origin of the essay, with 0 representing human-written essays and 1 indicating essays generated by an LLM. This clear, binary labeling simplifies the training objective for supervised learning models. Challenge: Due to the relative cleanliness of the dataset and the simplicity of the classification problem itself, it is possible to achieve up to 99% accuracy using only Language Model (LM) embeddings. While this phenomenon illustrates, on the one hand, the efficient ability of modern language models to understand and process natural language, it also poses the challenge of evaluating the performance advantages and disadvantages of different models.

3.1.2 Wiki Dataset

In our project, we leveraged the "GPT-wiki-intro Dataset" to develop models capable of distinguishing between text generated by humans and GPT-based models. This dataset comprises 150,000 entries, split evenly between human-written and GPT-generated introductions for a wide array of Wikipedia topics. Generated using the Curie version of GPT, the entries were crafted following a specific prompt that combines a topic's title with the initial seven words of its Wikipedia introduction. This design ensures a focused comparison between human and AI writing styles within a structured and consistent format. The dataset is rich with metadata, including the length of titles and introductions, the number of words and tokens in both the wiki and generated introductions, and the specifics of the prompts used for generation. This comprehensive data structure not only facilitates a deep analysis of linguistic patterns and differences between human and AI-generated texts but also supports the development of finely tuned detection models. Hosted on Hugging Face and available under a Creative Commons license, the "GPT-wiki-intro Dataset" is a pivotal resource for our project. It not only addresses the immediate need for sophisticated detection mechanisms in the wake of the growing use of LLMs but also sets a benchmark for dataset creation and utilization in AI research. This dataset has allowed us to explore the nuances of AI-generated text, contributing significantly to our project's success in developing robust models for detecting GPT-generated content.

3.2 Fixed Text Embedding

Word2vec:
Explanation: Word2Vec is a model that generates word embeddings, converting words into vector representations that capture semantic meanings. It operates through two architectures: CBOW, predicting a word based on context, and Skip-Gram, predicting context words from a target word. Word2Vec is effective for capturing linguistic relationships, useful for a variety of natural language processing tasks.
Implementation: We use word2vec package.

Glove:
Explanation: GloVe stands for Global Vectors for Word Representation. It generates word embeddings by analyzing word co-occurrences across the entire corpus, focusing on global statistics rather than local context. By factoring a word-context co-occurrence matrix, GloVe captures both semantic and syntactic word relationships efficiently.
Implementation: We use glove package.

Language model (fixed parameters):
Explanation: Language models with fixed parameters, like BERT, generate context-dependent word embeddings by considering the entire context around a word. Unlike Word2Vec and GloVe, these models provide embeddings that vary with the word's usage in text, enabling a deeper understanding of linguistic nuances for complex language tasks.

Implementation: We use three BERT-like methods, BERT, DeBERTa and DistilBERT. We utilized the package provided by hugging-face.

3.3 Classification

3.3.1 Machine Learning Methods

Logistic Regression:
Explanation: Logistic Regression is a statistical model used for binary classification tasks. It estimates probabilities using a logistic function, which outputs values between 0 and 1. This model is particularly useful for scenarios where you need to understand the influence of one or more independent variables on a binary outcome. Logistic regression is straightforward to implement, interpret, and very efficient for linearly separable classes. It serves as a foundation for more complex algorithms in machine learning and is widely used in fields like medicine, economics, and social sciences for predictive analysis.
Implementation: We use sklearn.linear_model.LogisticRegression with default hyperparameter settings.

Random Forest:
Explanation: Random Forest is an ensemble learning technique used for both classification and regression tasks. It builds upon the simplicity of decision trees by creating a forest of them, randomized through the selection of data points and features. This approach significantly reduces the risk of overfitting, common to individual decision trees, making Random Forest a more accurate and robust predictor. It excels in handling large datasets with higher dimensionality and can model complex relationships without requiring extensive data preprocessing.
Implementation: We use sklearn.ensemble.RandomForestClassifier with default hyperparameter settings.

Xgboost:
Explanation: XGBoost (Extreme Gradient Boosting) is an advanced implementation of gradient boosting algorithm, known for its speed and efficiency. It is designed to be highly efficient, flexible, and portable. XGBoost improves upon the basic gradient boosting framework through system optimization and algorithm enhancements, including handling missing values and regularizing to prevent overfitting. This model is capable of performing both classification and regression tasks and has gained popularity in machine learning competitions for its performance in handling a wide range of data types and tasks.
Implementation: We use xgboost package for implementation and conduct hyperparameter tuning (bayesian optimization) with Hyperopt. The fixed hyperparameters include objective ('binary:logistic'), and the tuned hyperparameters include booster, colsample_bynode, colsample_bytree, eta, gamma, lambda, rate_drop, subsample.

3.3.2 Deep Learning Methods

MLP using fixed embedding:
Explanation: An MLP with fixed embeddings is a neural network model for tasks like classification and regression, utilizing pre-trained vector representations of input features that remain static during training. The MLP structure includes multiple layers of interconnected neurons, learning complex relationships between input features and outcomes. Fixed embeddings provide a deep, albeit static, understanding of data semantics, aiding the MLP in efficiently mapping these inputs to target outputs without the need for learning feature representations from scratch.
Implementation: We use PyTorch.

LM Finetune:
Explanation: LM Fine-tuning adjusts a pre-trained language model to a specific task, enhancing its performance by slightly modifying the model's weights to better capture task-specific nuances. This approach leverages the comprehensive linguistic understanding from the model's initial training on extensive text data, requiring less data and time than training anew. Fine-tuned models excel in a variety of linguistic tasks, combining general language proficiency with specialized task knowledge.
Implementation: We use three BERT-like methods, BERT, DeBERTa and DistilBERT. We utilized the package provided by hugging-face.

Table 1: Fixed Text Embeddings for Kaggle Dataset

Embedding Type	Word2Vec	GloVe	BERT	DeBERTa	DistilBERT
Logistic Regression	98.09	96.03	99.43	98.35	95.52
Random Forest	98.19	97.23	98.77	98.81	98.73
XGBoost	98.88	98.06	99.28	99.37	99.19
MLP	99.07	97.01	98.81	98.55	99.14

Table 2: Fixed Text Embeddings for Wiki Dataset

Embedding Type	Word2Vec	GloVe	BERT	DeBERTa	DistilBERT
Logistic Regression	85.27	74.17	96.42	94.87	96.18
Random Forest	87.89	83.77	91.24	93.37	90.03
XGBoost	89.74	84.43	93.5	95.74	92.72
MLP	92.69	86.02	97.89	96.45	97.46

4 Results and Discussion

For all the results below, we utilize accuracy as the metric. We randomly split the dataset into training set and testing set.

4.1 Fixed Text Embeddings

4.1.1 Kaggle Dataset Experiment

The results can be seen in table 1. The small dataset appears to be relatively easy, exhibiting clear, distinct patterns among different classes. All embedding-model combinations have achieved accuracies of over 95%. Generally, dynamic embeddings generated from language models better capture information, resulting in higher classification accuracies. In the case of static embeddings like Word2Vec and GloVe, the more complex models, XGBoost and Random Forest, outperform Logistic Regression. Surprisingly, for dynamic embeddings, Logistic Regression can even outperform XGBoost and Random Forest. This phenomenon may be attributed to dynamic embeddings being trained in neural networks to preserve nonlinear information, hence requiring only a simple linear model for the final classification step.

4.1.2 Wiki Dataset Experiment

The results can be seen in table 2. The large dataset proves to be significantly more challenging, leading to a notable decrease in performance for static embeddings and a slight decrease for dynamic embeddings comparing with the Kaggle dataset. Across all three machine learning models, Word2Vec embeddings consistently achieve higher accuracies than GloVe, suggesting that Word2Vec is better suited for AI-generated content detection tasks. This observation may stem from Word2Vec's training methodology, which involves predicting target words based on context, aligning with the fundamental approach to machine-generated text. In contrast, GloVe relies on word co-occurrence matrices, which exhibit similarities between machine and human-generated text. As with the Kaggle dataset, Logistic Regression performs unsatisfactorily on static embeddings but surprisingly well on dynamic embeddings. However, the advantage of using Multilayer Perceptrons (MLP) for the final training stage is significant in this more demanding task, indicating that in domains where interpretability is less critical, neural networks remain a primary consideration.

4.2 Finetuning Language Model

Apart from those methods using fixed text embeddings, we also attempt to utilize the power of language model finetuning. In general, these methods gave us the best results from all of our experiments.

4.2.1 Kaggle Dataset Experiment

Kaggle Dataset's results can be seen in table 3. The "freeze parameters" setting is the same with using fixed language model's embeddings with MLP. As we can see, all three model's finetuning process

Table 3: Finetuning Language Models for Kaggle Dataset

Model Type	BERT	DeBERTa	DistilBERT
Finetune	99.68	99.58	99.47
Freeze Parameters	98.81	98.55	99.14

Table 4: Finetuning Language Models for Wiki Dataset

Model Type	BERT	DeBERTa	DistilBERT
Finetune	98.23	97.18	97.82
Freeze Parameters	97.89	96.45	97.46

did give us better results. However, because freezing parameters also gave us accuracies that aproach 100%, the improvement of finetuning is not significant.

4.2.2 Wiki Dataset Experiment

Wiki Dataset's results can be seen in table 4. The improvement is very similar comparing to Kaggle Dataset.

4.2.3 Influence of Training-Set Fraction

Here, we conducted a sensitivity study of finetuning the language models. We want to know whether different training set size will cause changes in the results. We tried training fraction from 0.1 to 0.9. As we can see in the figure 1, we found that a trend of increasing is what we expected. However, the increasing is very slight. Even when we took only 10% of the data to train the model, the results are very great. This suggests the embeddings given by language models are good enough, so that we can only use a small fraction of data to guide the embeddings into our specific task.

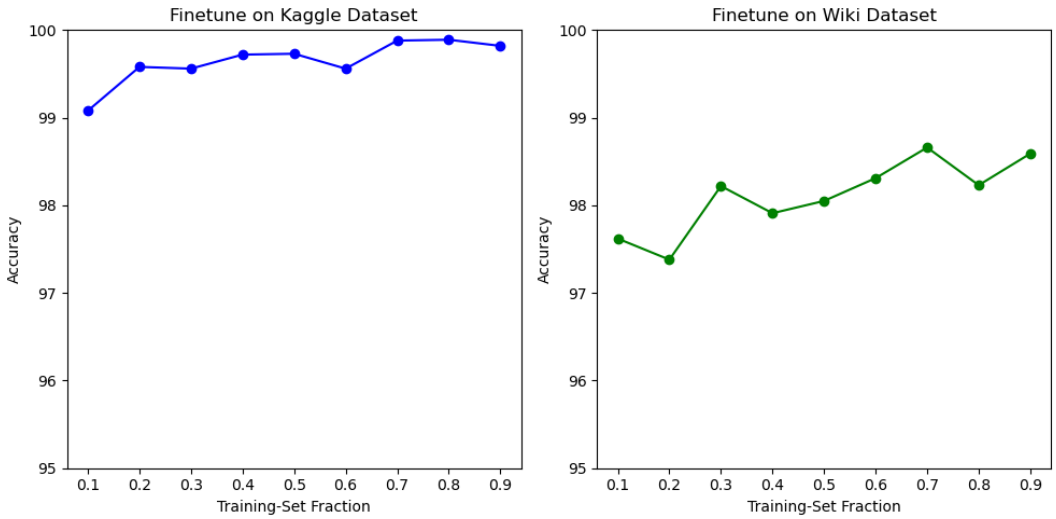

Figure 1: Influence of Training-Set Fraction

4.3 Visualization of Embeddings

To better understand the performances of those embeddings, we use TSNE to reduce the dimension of the embeddings, giving a 2-D representation. We can see the results of Kaggle Dataset in figure 2. We can see that all three methods (word2vec, GLoVe and BERT) gave a clear split between two types of nodes. Specifically, BERT methods gave us a linear separable representations. Thus, the results for BERT (and BERT-like) methods are the best between those fixed embeddings.

Table 5: Topic name for clusters

Cluster	Red	Green	Brown
Topic	Presidency	Exploring Venus	Scientists and Martian Research

Also, we did a case study for texts' topics. The intuition behind this is word2vec not only separate two types of nodes, but also forms tight node clusters in the representation. Therefore, we chose three cluster from word2vec embeddings. We used LDA to generate the potential topics of the three clusters, and conclude the exact topic name by ourselves. The three clusters forms three explicit topics. The topic names can be seen in table 5.

In this case, word2vec and GLoVe did much better than BERT. They gave more accurate clustering results while BERT produce overlapping between the topics. This may be because word2vec and GLoVe are more sensitive to concrete words and their relationships. On the contrary, BERT (and BERT-like methods) use deeper networks to get deeper understanding of texts.

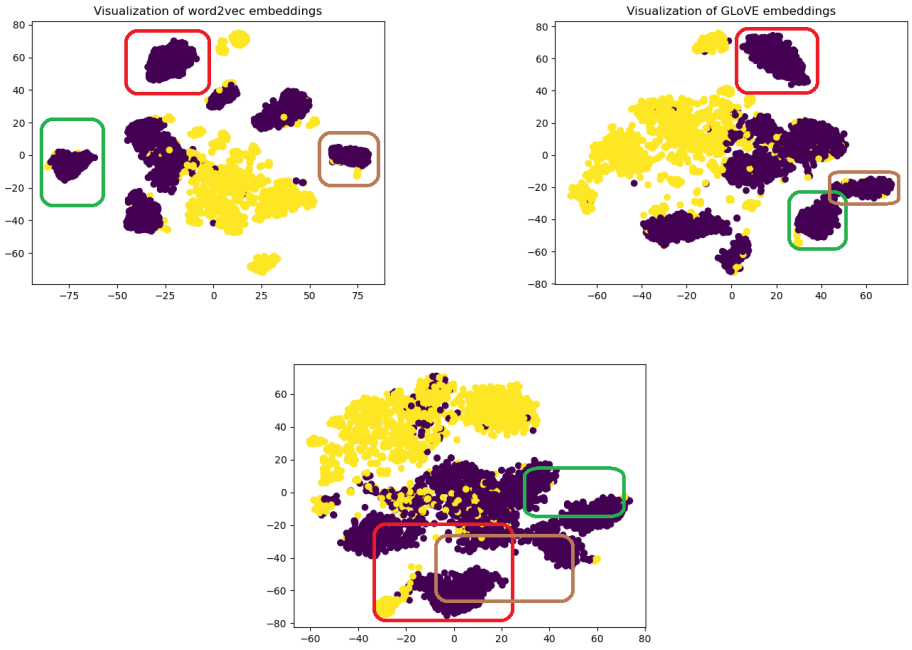

Figure 2: Embeddings for Kaggle Dataset

5 Conclusion

In this project, we explored the task of AI-generated text detection. We use a small dataset from Kaggle and a large dataset generated by GPT, using Wikipedia's related content. We set up a baseline and evaluate different text embeddings (word2vec, GLoVe, and BERT-like methods) using Machine Learning methods (Logistic Regression, XGBoost, and Random Forest) and Deep Learning methods (MLP). The key finding is that, in the task of AI-generated text detection, using a language model (BERT-like methods) to encode the texts and vanilla MLP for classification produces the best results. However, static encoding methods like word2vec and GLoVe also gave relatively good results considering that they don't need to load such big models like BERT. Also, we see the potential of language models by finetuning them and conducting a sensitivity study. Finetuning may take a relatively long time, but it can do much better if the task is more complicated. Last but not least, we visualized the embeddings to comprehend the task better and conducted a case study to further analyze the clustering effect on topics between each method.

References

[1] Chen, T., & Guestrin, C. (2016). XGBoost. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. https://doi.org/10.1145/2939672.2939785

[2] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Bidirectional Encoder Representations from Transformers. arXiv preprint arXiv:1810.04805.

[3] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

[4] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. Advances in Neural Information Processing Systems 30 (NIPS 2017). https://hal.science/hal-03953007

[5] Mikolov, Tomas; et al. (2013). "Efficient Estimation of Word Representations in Vector Space". arXiv:1301.3781.

[6] Mikolov, Tomas; Sutskever, Ilya; Chen, Kai; Corrado, Greg S.; Dean, Jeff (2013). Distributed representations of words and phrases and their compositionality. Advances in Neural Information Processing Systems. arXiv:1310.4546. Bibcode:2013arXiv1310.4546M.

[7] Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training. OpenAI Technical Report.

[8] Hosmer Jr, D.W., Lemeshow, S., Sturdivant, R.X. (2013). Applied Logistic Regression. John Wiley Sons.

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

6 Supplementary Material

6.1 Visualization of Embeddings (Wiki Dataset)

We also visualized the embeddings used for Wiki Dataset. However, the results is not that good comparing with Kaggle Dataset. We can see that the two types of nodes are mixed up together in the space. One reasonable guess is that the Wiki Dataset has much more noise, also possessing high dimensional structure, thus we may find the results obscure.

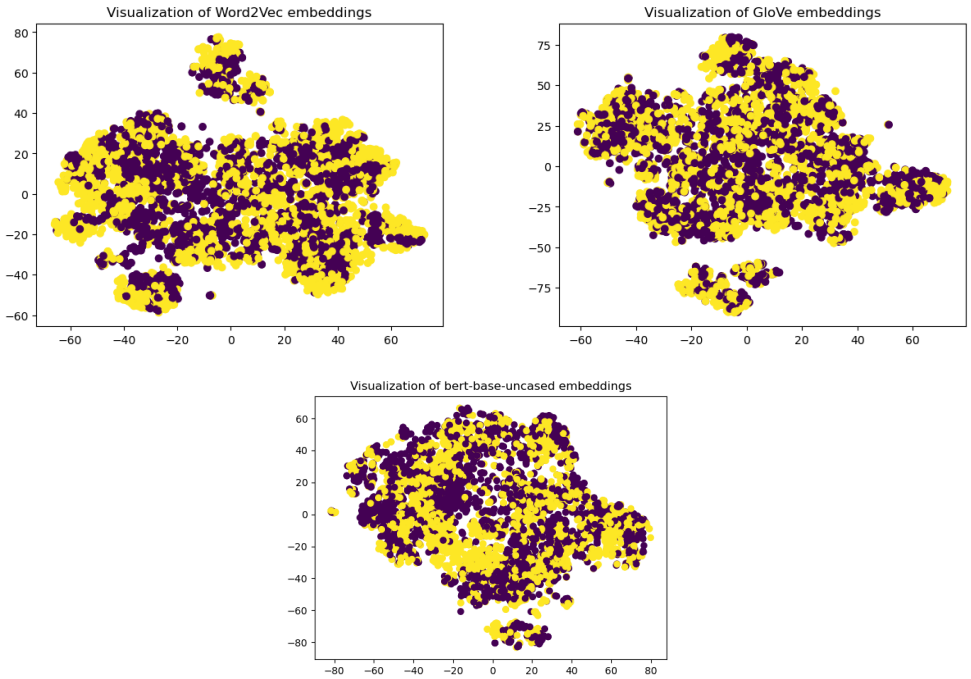

Figure 3: Embeddings for Kaggle Dataset