Regression with Text Features: an Experimental Approach Evaluated on Real-World Applications

Lucas O. Martinez
lom2017@nyu.edu

Alex Peterson
acp426@nyu.edu

Tapan Khaladkar
tk3301@nyu.edu

Abstract

Textual data is widely used in classification tasks, yet its potential for regression remains underexplored. This study examines traditional and embedding-based approaches to regression using text-derived features, applied to two real-world datasets: Supreme Court decisions for predicting decision years and LinkedIn job descriptions for forecasting salaries. Classical methods, such as TF-IDF and bag-of-words, are compared with embeddings from large language models like SBERT and T5. Regression models, including SVM (or SVR), MLP, and Gradient Boosting are evaluated using metrics like R^2, Root Mean Squared Error, and Mean Absolute Error. Notably, the Supreme Court dataset achieved an MAE of $\sim$12 years with Bigram BoW and $\sim$14 years with Doc2Vec embeddings. However, challenges remain, as demonstrated by the LinkedIn dataset, where the best result had an MAE exceeding \$160,000 USD for salary prediction. We hope that our findings highlight the promise and limitations of text-driven regression across real-world applications.

1 Introduction

The use of textual data in machine learning has been widely studied in classification tasks [1], where textual features are leveraged to categorize documents, detect sentiment, or perform topic modeling. However, the potential of textual data for regression tasks remains underexplored. Regression tasks involve predicting continuous values, and require a strong understanding of textual information to map it onto numeric values. This study seeks to address this gap by investigating if and how textual features can be effectively utilized for regression tasks across different domains.

Traditional approaches to text representation, such as Bag-of-Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF), offer interpretable and computationally efficient [2] methods for transforming text into numeric features. Despite their widespread use in classification [3][4][5][6], these methods have seen limited application in regression, where continuous predictions often require richer and more refined feature representations. By evaluating these classical techniques in regression settings, this research explores their adaptability to numeric prediction problems.

With the arrival of pre-trained large language models (LLMs) and embedding-based approaches, there has been a paradigm shift in how textual data is represented and used. Embeddings generated by models like T5 and BERT encode semantic and syntactic information into dense vectors, capturing intricate relationships within the text. Recent studies, such as those of Tang et al. (2024) [7] and Nguyen et al. (2024) [8], have shown that these embeddings can be used effectively for some regression tasks. This study builds on these and more insights by comparing embedding-based methods with traditional approaches to assess their suitability for regression tasks in practical real-world applications.

To ground our investigation, we focus on two datasets representing distinct challenges: (1) the LinkedIn Job Postings dataset [9], which requires the prediction of maximum salary based on job

descriptions, and (2) the Supreme Court Decisions (SCOTUS) dataset [10], which involves estimating the year of a legal decision based on textual content.

2 Related Work

To contextualize this research, we review related work on classical text representations and embedding-based approaches for regression tasks, highlighting existing methodologies.

As declared earlier, most text-based machine learning applications focus on classification tasks, often using complex deep learning models for feature extraction, such as transformers and embeddings. However, regression tasks using textual data remain relatively underexplored, particularly in the context of classical NLP approaches. Studies on feature extraction techniques such as Bag-of-Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF) are usually centered on classification tasks [11][12][13], leaving gaps in understanding their applicability for continuous prediction problems. Smoothing techniques for BoW, such as Laplace smoothing and add-k smoothing, have been applied in probabilistic models [14][15] but have seen limited exploration in non-probabilistic regression tasks, such as ridge regression or gradient-boosted decision trees.

Doc2Vec, introduced as an extension of Word2Vec [16], generates dense vector representations of entire documents rather than individual words [17]. This method has shown promise in capturing semantic and syntactic relationships within text, making it a natural candidate for tasks requiring document-level understanding [18]. While extensively used in classification and clustering, its potential for regression tasks remains underexplored. By encoding textual data into fixed-length representations, Doc2Vec facilitates regression analysis, particularly in applications where document context plays a critical role.

In addition, recent efforts by Tang et al. (2024) [7] and Nguyen et al. (2024) [8] have demonstrated the potential of embedding-based approaches for regression tasks. These studies highlight that embeddings derived from large language models (LLMs) capture rich semantic information, making them well-suited for numeric prediction tasks. Tang et al. [7] showed that embeddings from models such as T5 and Gemini provide robust representations, outperforming traditional hand-crafted features in certain regression scenarios. Nguyen et al. [8] proposed the "Embed-then-Regress" framework, which utilizes embeddings to map textual inputs to fixed-length vectors, enabling flexible and effective regression across diverse domains, including combinatorial optimization and hyperparameter tuning. Although these studies offer valuable insights, their publication occurred relatively late in our project's timeline, limiting our ability to fully explore their proposed methodologies. Nevertheless, these insights directly influenced our approach to feature selection, particularly the inclusion of T5 embeddings.

These advancements highlight the growing importance of embeddings in regression tasks. Building on insights from both classical and embedding-based approaches, our study evaluates their effectiveness by applying these methods to two practical datasets. This allows us to assess their performance and explore their applicability in real-world scenarios.

3 Method

To effectively represent textual data for regression tasks, we employed a diverse set of feature extraction methods, including traditional count-based techniques such as Bag-of-Words (BoW) and TF-IDF, as well as modern embedding-based approaches such as Doc2Vec, SBERT all-MiniLM, Paraphrase-MiniLM, and T5. Each method was selected for its ability to capture different and relevant aspects of textual information, as detailed in the following subsections.

3.1 Traditional Text Representation Techniques

3.1.1 Bag of Words (Unigram and Bigram Models)

The Bag of Words model is one of the simplest and most widely used techniques for feature extraction in text classification. It represents a text as a collection of words, disregarding grammar and word order while preserving frequency information. Each unique word or token is treated as a feature, and the text is encoded as a sparse vector with counts of each word. In unigram models, features

correspond to single words, while bigram models consider sequences of two consecutive words, capturing some contextual information. Thus, the inclusion of bigrams can improve performance in tasks where word order or phrase structure is important.

Unigram and bigram BoW models have been used extensively in text classification tasks. For instance, Pang et al. [3] demonstrated their utility in sentiment classification, showing that bigram models often outperform unigrams in capturing contextual nuances. Similarly, Joachims [4] used unigram features for text categorization with Support Vector Machines, highlighting the effectiveness of sparse high-dimensional representations in classification.

3.1.2 TF-IDF (Term Frequency-Inverse Document Frequency)

TF-IDF builds upon the Bag of Words model by weighting each term based on its frequency in the document (term frequency, TF) and its rarity across the corpus (inverse document frequency, IDF)[19]. This weighting scheme helps reduce the impact of common words (e.g., stop words) that are less informative for distinguishing between classes. The TF-IDF score for a term t in a document d is defined as:

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \cdot \text{IDF}(t),$$

where

$$\text{IDF}(t) = \log \frac{N}{1 + n_t},$$

N is the total number of documents in the corpus, and n_t is the number of documents containing term t.

TF-IDF has proven effective in numerous text classification applications. For example, Sebastiani [5] used TF-IDF to improve performance in automated text categorization, demonstrating its ability to highlight discriminative terms. Additionally, Lewis [6] employed TF-IDF features in Naive Bayes classifiers, achieving robust results in spam detection and topic classification.

3.2 Embeddings

3.2.1 Doc2Vec

Doc2Vec [20], an extension of Word2Vec, generates dense vector representations of entire documents instead of individual words [17]. By training the model to predict words in a context or paragraph based on document-level representations, Doc2Vec captures both semantic and syntactic information [18], making it a suitable choice for tasks requiring an understanding of the overall document. It also generates compact fixed-length vectors, facilitating the use of regression models.

3.2.2 SBERT (Sentence-BERT)

SBERT [21] is a modification of the BERT architecture fine-tuned to generate high-quality sentence embeddings. It employs a Siamese or triplet network structure to compare sentence pairs during training, optimizing for semantic similarity tasks using contrastive or triplet loss. SBERT is highly efficient in producing embeddings, making it well-suited for applications such as sentence similarity, semantic search, clustering, and textual entailment. Its embeddings are specifically designed for capturing semantic similarity, enabling rapid comparisons using cosine similarity. However, SBERT's flexibility is somewhat limited outside its fine-tuned domain, and the contextual quality of its embeddings may degrade for sentences with significantly different lengths or structures. For our experiments, we worked with the all-MiniLM-L6-v2 SBERT model.

3.2.3 Paraphrase-Based Embeddings

Paraphrase-based embeddings are generated using transformer-based models, such as BERT or RoBERTa, trained specifically on paraphrase datasets. These models are optimized to detect whether two sentences are paraphrases, ensuring that embeddings for similar sentences are closer in the vector space. Paraphrase-based embeddings are highly effective for tasks like paraphrase detection and similar sentence retrieval, excelling at identifying nuanced rephrasings or logical equivalences. However, their general-purpose applicability is more limited compared to SBERT or T5, as their performance is closely tied to the quality of the paraphrase datasets used for training.

3.2.4 T5 (Text-to-Text Transfer Transformer)

The T5 model [22] is a versatile sequence-to-sequence model that treats all NLP tasks as a "text-to-text" problem, capable of generating embeddings as a by-product of its encoding process. While its primary design is not for embedding generation, the embeddings produced by T5 are contextually rich due to pretraining on a diverse text corpus using a denoising objective. T5 is highly flexible and can be fine-tuned [23] for a wide range of NLP tasks, such as text summarization, question answering, and more. However, it is computationally expensive compared to SBERT, and its embeddings may require additional fine-tuning to achieve similar alignment for semantic similarity tasks. Despite this, T5's generalization capabilities make it a strong choice for diverse and generalized NLP tasks.

3.3 Regression Models

We have worked with three regression models to evaluate the effectiveness of different textual feature extraction techniques. These models were chosen for their ability to handle diverse data distributions and capture both linear and non-linear relationships in the data.

For all the models described in this subsection, hyperparameter tuning was performed following a two-stage process:

1. **Random Search:** To broadly explore combinations of key hyperparameters.
2. **Grid Search:** To "fine-tune" the most promising combinations identified by the random search.

3.3.1 SVM (Support Vector Machine)

Support Vector Machine (SVM), often applied to both classification and regression tasks, seeks to find a hyperplane that minimizes error while balancing model complexity and prediction accuracy. We chose SVM (SVR for regression) because of its effectiveness with high-dimensional, sparse text features (such as those generated by Bag-of-Words and TF-IDF), and its flexibility in capturing both linear and non-linear relationships through kernel functions [24].

We experimented with multiple kernel types, including linear and radial basis function (RBF) kernels, to accommodate the high-dimensional and sparse nature of text features while also exploring potential non-linear relationships. As mentioned, hyperparameter tuning was conducted in two stages tuning the following: regularization parameter C, margin of tolerance ϵ, and Kernel-specific parameters (e.g., Linear Kernel, RBF, gamma).

3.3.2 MLP (MultiLayer Perceptron)

The MultiLayer Perceptron (MLP) was selected as a regression model for its ability to capture non-linear relationships in high-dimensional data. The decision to also experiment with MLP was inspired by Tang et al. (2024) [7], which highlights the suitability of MLPs for processing high-dimensional embeddings derived from textual data. Their ability to model non-linear relationships and their compatibility with embedding-based feature spaces make them a strong candidate for regression tasks.

The architecture of the MLP used in our experiments is as follows:

- Input layer: Accepts the feature vector of dimensionality corresponding to the extracted text features.
- Two hidden layers: The first layer contains 128 neurons, and the second layer has 64 neurons. Both use ReLU activations to introduce non-linearity.
- Dropout layer: A dropout rate of 0.3 is applied after the first hidden layer to prevent overfitting.
- Output layer: A single neuron outputs the predicted value.

As with the previous regressor, we performed parameter tuning on the following hyperparameters: learning rate, dropout rate, hidden layer sizes.

3.3.3 Gradient Boosting

Gradient Boosting is an implementation of gradient-boosted decision trees designed for speed and performance. It builds an ensemble of weak learners (decision trees) iteratively to minimize a specified loss function. Gradient Boosting was chosen based on its demonstrated effectiveness in regression tasks involving high-dimensional data [7], as well as it's ability to handle complex relationships.

We used Gradient Boosting's regression module and tuned the following hyperparameters: learning rate, max depth, subsampling rate.

4 Dataset

Having established the embedding models and regression techniques, we now introduce the datasets used to test these methods, highlighting their distinct challenges and applications.

Our study utilizes two distinct datasets, the first one, the LinkedIn Job Postings dataset [9], provides a practical real-world scenario for salary forecasting. This model can assist job seekers in estimating potential salaries before engaging in lengthy and tedious application and interview processes. It can also help employers to properly gauge the competitiveness of their compensation.

The second one, the Supreme Court Decisions (SCOTUS) dataset [10], offers an opportunity to explore the temporal evolution of formal legal language. Additionally, it demonstrates a potential practical application for historians or researchers in estimating the year of authorship based on old, partially preserved documents in a wide variety of domains.

4.1 LinkedIn Job Postings Dataset

The LinkedIn Job Postings dataset [9], sourced from Kaggle, contains a collection of job postings that includes detailed textual and metadata fields. This dataset provides a robust foundation for analyzing the linguistic and contextual attributes of professional job descriptions, with the aim of predicting salary information.

4.1.1 Content

Each record in the dataset includes the job description, title, and location, alongside additional metadata fields such as company name and employment type. The textual content of job descriptions captures domain-specific language, skills requirements, and employment terms. Some data cleaning was required before modeling could begin, including annualizing hourly or monthly salaries and removing missing values.

4.1.2 Modeling Task

The predictive modeling task focuses on estimating the minimum and maximum salary for each job posting. The model leverages textual features (e.g., key phrases, industry-specific terminology) and contextual variables (e.g., job title, geographic location) to infer these continuous numeric targets.

4.1.3 Challenges

The dataset exhibits variability in text length, structure, and specificity, reflecting the diversity of industries and roles. It also suffers from an uneven distribution of salary ranges across industries and geographic locations. However, we hypothesized that there would be some signal between natural language descriptions of tasks and responsibilities and the numerical value of annual salary.

4.2 Supreme Court Decisions Dataset

The Supreme Court Decisions dataset [10], hosted on GitHub as part of the SCOTUS project, is a compilation of text and metadata from historical U.S. Supreme Court decisions. This dataset is particularly valuable for analyzing legal language and its evolution over time.

4.2.1 Content

Each record consists of the full text of a Supreme Court opinion, along with metadata fields such as the case name, decision date, and authoring justice. The textual data reflects formal legal reasoning, structured argumentation, and domain-specific terminology.

4.2.2 Modeling Task

The primary task involves predicting the year of the decision based on the text of the opinion. This task captures the temporal evolution of legal language and reasoning, necessitating the model's ability to discern subtle linguistic patterns that correlate with the period of authorship.

4.2.3 Challenges

The dataset presents unique challenges, such as the presence of semantic drift, where the meaning and usage of certain legal terms change over time. Additionally, imbalances in the distribution of decisions across years require careful handling during training and evaluation. Feature extraction techniques, such as bag-of-words and n-grams, are instrumental in encoding the textual data, while smoothing techniques address sparsity in the high-dimensional feature space.

4.3 Comparative Utility

Both datasets offer distinct yet complementary challenges for text-based regression modeling. The LinkedIn Job Postings dataset emphasizes practical applications in economic forecasting, while the Supreme Court Decisions dataset provides insights into temporal patterns in formal legal discourse. Together, they enable a comprehensive evaluation of classical regression techniques in diverse real-world and domain-specific contexts.

5 Experiments

The experiments were designed to evaluate the performance of various text feature extraction techniques combined with regression models on two datasets: (1) the LinkedIn Job Postings dataset, where the target variable is the maximum salary, and (2) the Supreme Court Decisions dataset, where the target variable is the year of the decision. To assess performance, we'll use the following metrics: R^2, Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE)

5.1 LinkedIn

For the LinkedIn dataset, we explored different combinations of text features to identify the most informative set for predicting maximum salary. We experimented with the following combinations:

- (title + description + location)
- description
- title
- (title + description)

Our initial findings indicated that embedding the job title alone using the T5 model gave the best performance in terms of RMSE. Table 1 shows the RMSE scores obtained by applying embeddings and text vectorization to the combination of features: (title + description + location) followed by training three different regression models on the resulting featurized data. Note that t5_total_embedding refers to applying the T5 model for embedding to feature (title + description + location), and t5_title_embedding refers to applying the T5 model for embedding to the feature "title". The same applies for Paraphrase embedding.

To maintain a concise and focused analysis, we present in Figure 1 the detailed results of using *Gradient Boost*. We chose to show only the results for this regression model because, as seen in Table 1, this is the model that achieved the lowest RMSE.

Although the results are still quite poor, it is interesting that embeddings derived from job title alone outperformed the concatenation of title, description and location. This implies that the job

Table 1: RMSE scores for different text vectorization and embeddings models for Linkedin dataset when using SVM (i.e. SVR), MLP, and Gradient Boosting regressor models

Model Name	SVM	MLP	GBoost
bigram	3,621,106	9,176,577	5,270,339
t5_title_embedding	7,126,253	4,369,613	**3,311,473**
paraphrase_title_embedding	7,708,288	6,853,952	7,207,586
t5_total_embedding	4,520,921	3,962,114	12,414,734
smoothed_unigram	3,510,822	8,406,475	9,719,562
paraphrase_total_embedding	3,609,581	8,127,094	9,126,607
unigram	9,698,041	3,455,958	7,830,285
tf-idf	3,129,469	6,452,979	7,571,658
all-MiniLM-L6-v2	3,620,362	5,601,687	7,583,013
doc2vec	3,621,309	3,489,431	3,357,549

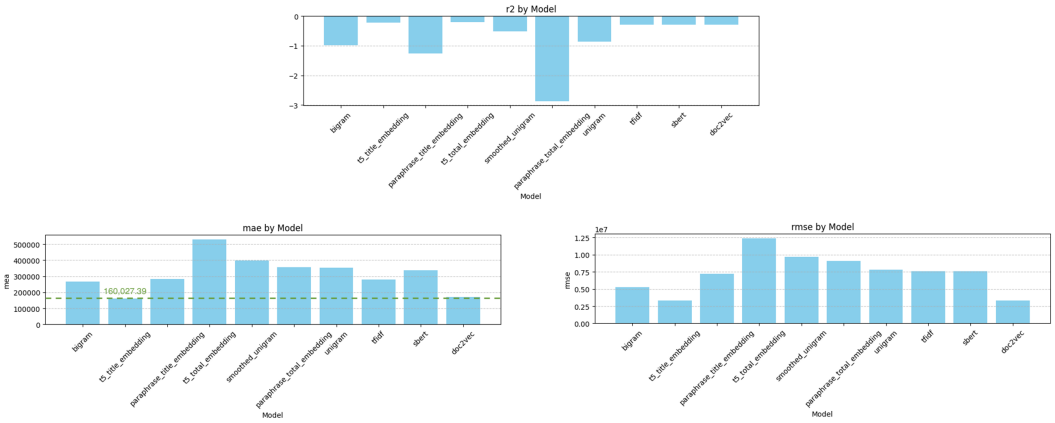

Figure 1: R2, MAE, and RMSE scores for Gradient Boost with the different embedding methods on the LinkedIn dataset

descriptions in this dataset were noisier than the one or two tokens in the titles. Indeed, a list of a job responsibilities might have a more tenuous relationship with salary compensation than words like "senior", "partner", "entry" or "technician". However, we can see that the selection of text does not completely eclipse the importance of the embedding model: the paraphrase embedding of all available text features (i.e. paraphrase_total_embedding) outperformed the corresponding t5 embedding in most cases, despite the fact that t5 provided the best embedding when applied to title alone.

5.2 SCOTUS

We applied a similar experimental framework to the SCOTUS dataset, adapting the feature extraction methods to its specific characteristics (i.e. embedding was applied only to the "text" feature) and exploring their impact on prediction performance. In Table 2 we can observe the RMSE of using multiple embedding techniques with SVM, MLP, and Gradient Boosting.

Again, to maintain a concise and focused analysis, we present in Figure 2 the detailed results of using *Gradient Boost*. As seen in Table 2, this is the regression model was the one that achieved the lowest RMSE.

The results from the SCOTUS dataset experiments reverse the conclusions of the linkedin dataset experiments: vectorization techniques relying on simple counts and ratios (like bigrams and TF-IDF) significantly outperformed more complex embedding models. We can hypothesize that the presence or absence of certain words has a much stronger relationship to the year that certain text was written then any other linguistic property. This follows some intuitive understanding of linguistic trends falling in and out of fashion as years progress. An area of future research might investigate whether this property holds in other domains like literature and journalism.

Table 2: RMSE scores for different text vectorization and embeddings models for SCOTUS dataset when using SVM (i.e. SVR), MLP, and Gradient Boosting regressor models

Model Name	SVM	MLP	GBoost
bigram	30.79	532.31	**18.38**
t5_embeddings	47.01	907.18	40.33
paraphrase_embedding	30.36	36.91	43.40
smoothed_unigram	48.40	247.62	22.27
unigram	33.20	543.08	20.40
tf-idf	26.78	53.55	22.68
all-MiniLM-L6-v2	48.92	456.47	41.76
doc2vec	39.70	296.91	24.64

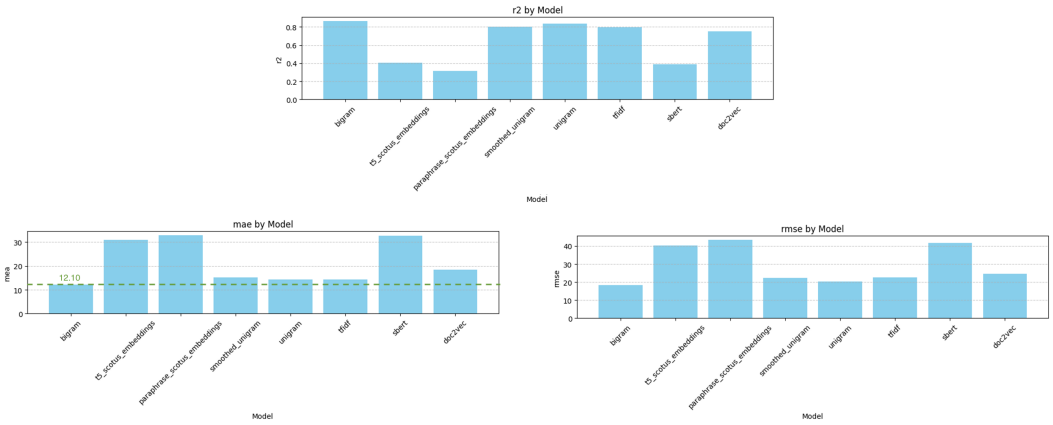

Figure 2: R2, MAE, and RMSE scores for Gradient Boost with the different embedding methods on the SCOTUS dataset

6 Conclusion and Future Work

This study explored regression models leveraging textual features to predict continuous outcomes across two datasets: the LinkedIn job postings dataset for salary prediction and the Supreme Court decisions dataset for year prediction. While the Supreme Court dataset yielded promising results, with a MAE of around 12 years using Bigram BoW (see Figure 2), the LinkedIn dataset faced challenges, including weak correlations between text and salaries. Despite embeddings like title-based T5 performing better (with an MAE exceeding $160,000 USD as shown in Figure 1), the LinkedIn dataset remained difficult to model probably due to inherent noise and skewness toward lower salaries ($22,000 USD and below).

Our findings emphasize that the success of text-driven regression is highly task-dependent. Count-based methods like BoW and TF-IDF excelled for legal language tasks, likely due to the stable and patterned nature of the text. And, as mention in the previous section, this aligns with the intuitive notion that linguistic trends evolve over time, with certain patterns and expressions gaining or losing popularity as years go by. In contrast, embedding-based approaches offered only limited improvements for complex datasets like LinkedIn, highlighting the need to carefully align feature extraction techniques with dataset characteristics.

While these findings provide valuable insights, they also highlight areas for improvement and further exploration. Future work should focus on refining dataset scope, such as narrowing the LinkedIn dataset to specific job sectors or salary ranges, to improve signal strength. Additionally, exploring more complex models, including deeper neural networks and fine-tuned embeddings, could enhance predictive power for both applications evaluated in this work. In addition, investigating alternative target variables, such as salary ranges (e.g., max-min), may also yield better insights and improve model interpretability for the LinkedIn dataset.

References

[1] Qian Li, Hao Peng, Jianxin Li, Congying Xia, Renyu Yang, Lichao Sun, Philip S. Yu, and Lifang He. A survey on text classification: From traditional to deep learning. *ACM Trans. Intell. Syst. Technol.*, 13(2), April 2022.

[2] Deepa Rani, Rajeev Kumar, and Naveen Chauhan. Study and comparision of vectorization techniques used in text classification. In *2022 13th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pages 1–6, 2022.

[3] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 79–86. Association for Computational Linguistics, July 2002.

[4] Thorsten Joachims. Text categorization with support vector machines: learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, ECML'98, page 137–142, Berlin, Heidelberg, 1998. Springer-Verlag.

[5] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, March 2002.

[6] David D. Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In *European Conference on Machine Learning*, 1998.

[7] Eric Tang, Bangding Yang, and Xingyou Song. Understanding llm embeddings for regression, 2024.

[8] Tung Nguyen, Qiuyi Zhang, Bangding Yang, Chansoo Lee, Jorg Bornschein, Yingjie Miao, Sagi Perel, Yutian Chen, and Xingyou Song. Predicting from strings: Language model embeddings for bayesian optimization, 2024.

[9] Linkedin job postings (2023 - 2024). `https://www.kaggle.com/datasets/arshkon/linkedin-job-postings`. Accessed: 2024-10-30.

[10] Collection of supreme court of the united states' opinions. `https://github.com/EmilHvitfeldt/scotus`. Accessed: 2024-10-30.

[11] Wisam A. Qader, Musa M. Ameen, and Bilal I. Ahmed. An overview of bag of words;importance, implementation, applications, and challenges. In *2019 International Engineering Conference (IEC)*, pages 200–204, 2019.

[12] Karen Spärck Jones. A statistical interpretation of term specificity and its application in retrieval. *J. Documentation*, 60:493–502, 2021.

[13] Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29, 1990.

[14] William A. Gale and Kenneth Ward Church. - 1-what ' s wrong with adding one ? 1994.

[15] S. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35(3):400–401, 1987.

[16] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[17] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents, 2014.

[18] Jey Han Lau and Timothy Baldwin. An empirical evaluation of doc2vec with practical insights into document embedding generation, 2016.

[19] Stephen Robertson. Understanding inverse document frequency: On theoretical arguments for idf. *Journal of Documentation - J DOC*, 60:503–520, 10 2004.

[20] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents, 2014.

[21] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019.

[22] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2023.

[23] Antonio Mastropaolo, Simone Scalabrino, Nathan Cooper, David Nader Palacio, Denys Poshyvanyk, Rocco Oliveto, and Gabriele Bavota. Studying the usage of text-to-text transfer transformer to support code-related tasks. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pages 336–347, 2021.

[24] M.A. Hearst, S.T. Dumais, E. Osuna, J. Platt, and B. Scholkopf. Support vector machines. *IEEE Intelligent Systems and their Applications*, 13(4):18–28, 1998.