Final Project Report

Assessing the Effectiveness of Linkedin Company Description in Classifying Industry

By Ya Yun Huang, Emma Wang, Heather Qiu

Professor: *Dr. Patrick Wang*

Duke University

Course: IDS 703 Natural Language Processing

Assignment: Final Project Report

Date: 18.December.2022

I. Project Overview

The primary objective of the project is to evaluate whether company descriptions on LinkedIn are sufficient to identify the type of business. We manually gathered organizational information from LinkedIn, including company names, profiles, and industries. The current study focuses on three predominant industries: Financial Services, Hospitals & Health Care, and IT Services & IT Consulting. However, our analysis can be scaled to incorporate additional business sectors for broader use cases. For instance, the language models employed in the research provide a good foundation for data applications that seek to generate sample company overviews and/or evaluate the quality of the generated company descriptions. The graph below visualizes popular keywords of LinkedIn organization profiles. The larger the size of the word, the more frequently it appears in the dataset.

Figure 1. *Popular LinkedIn Company Description Keywords*

II. Project Design

Our analysis involves four main steps:

1. ***Data Scraping***

   Given the research objective, we manually scraped LinkedIn organizational information in three aspects: company names, descriptions, and their corresponding type of business. The dataset includes a total of 2,705 unique entities on Linkedin across three leading industries in the United States. Because we performed a preliminary review when sourcing the data, the dataset contains only organizations with non-empty English company profiles. Overall, the dataset

is balanced among business sectors of interest. The sample size of each business sector is listed as follows:

Table 1. *Company Breakdown by Industry*

	IT Services and IT Consulting	Financial Services	Hospitals and Health Care
Count	833	926	946

2. *Data Processing*

The first step of data wrangling is to one-hot encode the three business sectors. As a result, Financial Services, Hospital & Health Care, and IT Services and IT Consulting industries are labeled as 0, 1, and 2, respectively. Next, we performed a series of text processing to improve the overall cleanliness of qualitative data input prior to modeling. The text-munging steps include the following:

- ➢ Removal of Unicode characters, leading or trailing spaces, and punctuations in company descriptions
- ➢ Conversion of company profiles to lower-case only
- ➢ Tokenization of company overview in string format to a list of words
- ➢ Elimination of common English stopwords
- ➢ Lemmatization using WordNet's built-in Morphy function
- ➢ Concatenation of lists of tokens back to individual sentences

Upon data cleaning, we generated individual word clouds for each industry of interest to gain additional insights into the top 10 company profile keywords (see figure 2).

Figure 2. *Top 10 Description Keywords of Each Industry*

3. Modeling on Original Data

3.1 Split Original data into training set and test set and Label encoding

We first randomly (with random seed = 1) split our data into a training set and a test set, which includes 80 % and 20% of our original data. As a result, the training set includes 2,164 data points, and the test set has 541 data points.

3.2 Training Original Data using a Generative Model

We used Multinomial Naive Bayes (NB) Model as the generative model to classify the three industries, Financial Services, Hospital & Healthcare, and IT Services & IT Consulting, respectively, based on their LinkedIn company descriptions.

Before fitting our texts into the Multinomial NB model, we used the CountVectorizer offered by the Scikit-learn library to represent our texts as features (Pedregosa, 1970). To be more specific, the CountVectorizer transforms each company description into a vector representation, denoting the count of a word in that text. There are 6937 unique words in the train set. Because we passed all of them through the CountVectorizer, the texts of the train set and test sets were transformed into matrices X-train and X-test of size (2164, 6937) and (541, 6937), respectively.

Next, we trained the Naïve Bayes model offered by the ScikitLearn package using X-train and its corresponding labels. The Multinomial NB classifier we used here calculates the probability of a class given a set of feature values. In other words, the model calculated $p(y_i \mid x_1, x_2, \ldots, x_n)$ where x_1-x_n represents each entry of the Count representation of each text. The Multinomial NB model assumes that all features – all words – are independent. Upon training the model, we predicted classes for the test set with X-test. The accuracy of the Multinomial NB model on the test set of the original data is 93%.

3.3 Training Original Data using a Discriminative Neural Network Model

We chose the Bidirectional Encoder Representations (BERT) model as our discriminative model in the research. More specifically, we adopted the 'bert-base-cased' model from HuggingFace and fine-tuned the model (Devlin *et al.*, 2019).

3. 3. 1 Pre-trained BERT Model objective tasks and Datasets

The BERT model from Hugging Face was initially pre-trained for two unsupervised tasks: masked ℵ language modeling(MLM) and next-sentence prediction(NSP). The original BERT was trained on BooksCorpus with 800 million words and Wikipedia with 2500 million words. In MLM, 15% of words in each sequence become masked

prior to feeding word sequences into BERT. The model attempts to predict the original value of the masked words based on the context provided by the non-masked tokens in the sequence. This task improves BERT's performance when dealing with unarranged or corrupted input. In NSP, the model receives pairs of sentences as input and learns to predict if the second sentence in the pair is the subsequent sentence in the original document. The NSP task allows BERT to learn relationships between sentences. When training the BERT model, MLM and NSP are trained together to minimize the combined loss function of the two strategies.

3. 3. 2 Pre-trained BERT Model Architecture

At its core, the BERT model consists of stacked bidirectional encoders from the transformer model, which can read all words in a text at once and therefore model their relationship.

➢ Input embedding

As shown in ***Figure 3***., the input embeddings for BERT are the sum of the token embeddings, the segmentation embeddings, and the position embeddings. The token embeddings are the pre-trained embedding of words. The segmentation embeddings are essentially the encodings of the input's "sentence" position into a vector. The position embeddings are encodings of the input position. Also, the first token before the first word would be initialized as [CLS], and the [SEP] tokens would be inserted as separations of sentences.

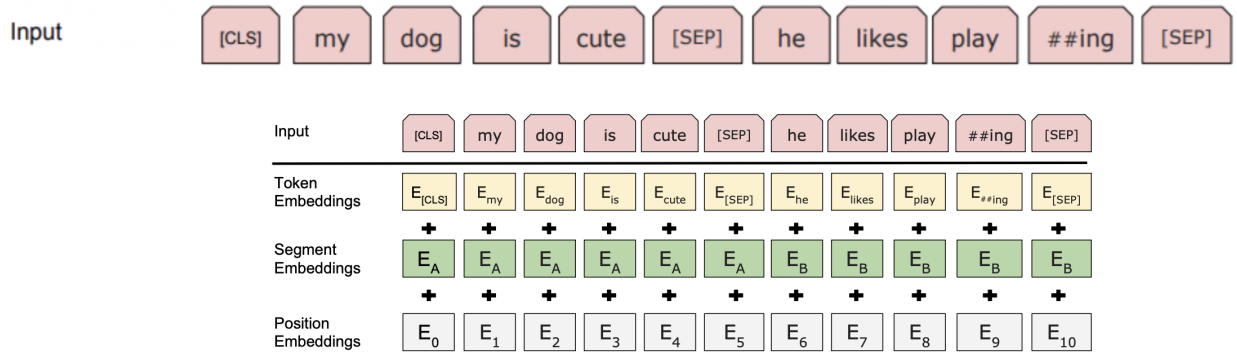

Figure 3. *Input embeddings [Devlin et al., 2018]*

➢ Model structure

As shown in ***Figure 4***, the embedded inputs would be fed into multiple encoders(or transformers, Trm) simultaneously and outputs of these encoders will be fed into the next layer of Trms. In each Trm, multiple attention heads capture the relationship of different inputs. Hidden size is defined as the size of each input token(word) in a sequence. For the model we used, there are 12 stacks of encoder layers, the hidden size is 768 and there are 12 attention heads in each Trm.

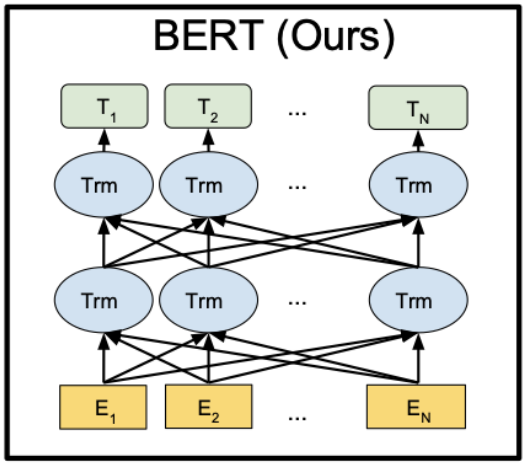

Figure 4. *Architecture of BERT [Devlin et al., 2018]*

For our task, we need a pre-trained BERT model for classification, which is different from the objective of the original BERT. Therefore, a classification layer including a linear and a softmax activation layer will be added on top of the [CLS] output as shown in ***Figure 5.***

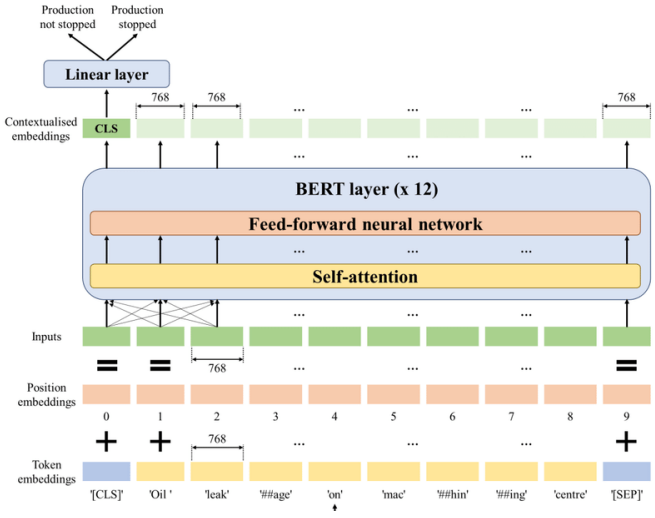

Figure 5. *Architecture of the BERT classifier (Segmentation embedding of input is not shown in this figure)[Cadavid et al., 2021]*

➢ Fine-tuning the BERT model

In order to directly compare the effectiveness of BERT with Multinomial NB, we used the same train set and test set as in the generative model. The key difference between the two models is that BERT has an embedded encoder mechanism with its

own tokenizer and only requires textual input (i.e., the train set). Texts of X-train and X-test were both transformed into vectors through a pre-trained tokenizer, *BertTokenizerFastf*. To fine-tune the BERT classifier, an extension of the classic stochastic gradient descent method, Adam optimization, was used. Adam optimizer leverages the power of adaptive learning rates methods to find individual learning rates for each parameter. The optimizer was set with the following hyperparameters: initial learning rate = 5e-5, decay steps = 1000, and decay rate = 0.9. The loss function used to optimize this optimizer is Cross Entropy Loss. Also, the batch size was set to 32. Then, our pre-trained BERT Model was fine-tuned with the training set for 1 epoch. After fine-tuning, the model was used to predict classes for tokenized texts of the test set. The accuracy of the fine-tuned BERT model on the test set of the original data is 89%.

4. Synthetic Data Generation and Training

4.1 Process of Generating Synthetic Data

To generate synthetic data, we first created a dictionary using the labels and their corresponding word distribution in empirical log probability, P(X|y), from the multinomial Naive Bayes model. Then, we used a NumPy random generator developed by Harris *et al.*(2020) to arbitrarily draw samples from the distributions and denote the desired size of synthetic datasets. For the synthetic data of each class, we drew 50 samples, indicating 50 words of the corresponding class distribution. For each generated text, we have a vector of count representations for all words in our vocabulary set, which includes 6937 words used in the previous CountVectorizer. Thus, the aforementioned step allows us to generate texts backward. We generated 300 synthetic texts for each class, and therefore we have 900 synthetic texts in total.

All synthetic texts are split into the training and the test set with a split ratio of 80:20. For Multinomial NB mode, synthetic texts for both sets were transformed into vectors using count representation as further inputs. For the BERT model, synthetic texts were encoded using a pre-trained *BertTokenizerFast* tokenizer.

4.2 Using Multinomial NB and BERT Models to Train and Test Synthetic Data

Both the Multinomial NB and the BERT models were trained and tested using the same way described in step 3.2 and 3.3 with the synthetic data.

- ➢ The accuracy of the multinomial naive bayes model on synthetic data is 100%.
- ➢ The accuracy of the multinomial naive bayes model on synthetic data is 98%.

III. Training Time/Computational Requirements

For the BERT model, we used the GPU core in GitHub CodeSpaces, which took roughly 50 minutes to complete the entire training process. On the other hand, the Multinomial NB model seemed much less computationally intensive, as the run time was within a reasonable timeframe (in seconds).

IV. Model Correctness/Interpretation

Overall, the generative and the discriminative models effectively predict the industry of a business based on their company description. Tables 2-4 show side-by-side comparisons of the accuracy, precision, recall, and f1-score between the two models.

Table 2. *Model Accuracy*

	Generative Model (Multinomial Naive Bayes)	Discriminative Model (BERT)
Accuracy on Original Data	93%	89%
Accuracy on Synthetic Data	100%	98%

Table 3. *Model Precision, Recall, and F1-Score by Industry on original dataset*

	Industry	Precision	Recall	F1-Score
Generative Model (Multinomial Naive Bayes)	Financial Services	94%	91%	92%
	Hospital & Health Care	96%	93%	94%
	IT Services & IT Consulting	88%	94%	91%
Discriminative Model (BERT)	Financial Services	95%	90%	92%
	Hospital & Health Care	98%	83%	90%
	IT Services & IT Consulting	78%	95%	86%

Table 4. *Model Precision, Recall, and F1-Score by Industry on synthetic dataset*

	Industry	Precision	Recall	F1-Score
Generative Model (Multinomial Naive Bayes)	Financial Services	100%	100%	100%
	Hospital & Health Care	100%	100%	100%
	IT Services & IT Consulting	100%	100%	100%
Discriminative Model (BERT)	Financial Services	100%	95%	97%
	Hospital & Health Care	100%	98%	99%
	IT Services & IT Consulting	94%	100%	97%

Overall, both of the generative and discriminative models are already performing well on our original dataset, yielding 93% and 89% accuracy, respectively. The higher accuracy of the Multinomial Naive Bayes model is likely due to the model assumption of conditional independence being well met. In other words, the generative model, Naive Bayes in our case, can still estimate the probability by marginalizing the unseen vocabularies in the test set due to the assumption. When re-training the models on the synthetic data, the performance became even better, reaching 100% accuracy in Naive Bayes model and 98% accuracy in BERT. As such, we can see that both the generative and discriminative models perform better under the conditions which the Naive Bayes generative method was designed for than under the real conditions (explanation provided below).

The precision reflects how reliable the model is in classifying samples correctly, while the recall score demonstrates the model's ability to detect the class. When training models on the real dataset, the IT Services industry tends to have lower performance than the other two industries in both the BERT and the NB models. For example, it is not easy for BERT to identify the IT services industry when training on real data; a 78% precision score entails a 22% error that BERT wrongly classifies other industries as IT service class. It is also worth noting the 83% recall score in the Health industry, meaning 17% of health industry company descriptions were incorrectly labeled by BERT. We observed similar performance patterns of these three industries in the Naive Bayes model.

When it comes to synthetic data, both generative and discriminative models have much better performance. Probable reasons might derive from the data generation process. Given that we drew the samples from the probability distributions based on Naive Bayes, we are more likely to get the words that frequently appear considering the class. Thus,

uncommon vocabularies are less likely to surface in the synthetic dataset, which greatly improves the accuracy score.

V. Limitation

Data-wise

The first constraint is that only three industries were incorporated due to the labor-intensive data collection process. Adding additional business sectors in future studies will not only enable a more comprehensive assessment of the quality of Linkedin company descriptions but also help train robust models that can be used across many industries. The second limitation originates from the single data source, as our research currently leverages data solely from LinkedIn. While the platform provides a good source of information, we might be able to refine industry classification with an exhaustive corpus and thus improve model accuracy. The third constraint stems from the number of grams our tokenizer takes in. The current approach splits the company description word by word (i.e., one gram). While the unigram model proves to be a valid approach, loss of information is inevitable because we no longer consider the meaning of phrases.

Model-wise

The ***Naive Bayes model*** assumes the independence of all the features. In other words, each vocabulary in the company descriptions is independent of one another, and the order of words does not matter. However, this is rarely true in reality, given that the speech sequence is crucial in understanding the context. Fortunately, such a limitation does not weigh heavily on the model's performance. This is likely because we are classifying the industry based on company descriptions that do not contain many sentimental words or complex contexts. The model might already be able to specify the industry class based on the combination of a few keywords.

As for ***BERT*****,** we would like to note that it primarily fine-tunes tasks that involve the evaluation of whole sentences (potentially masked) to make decisions, such as sequence classification, token classification, or question answering. For analysis involving text generation, one should look into other models, for example, GPT2 (Radford et al., 2019). One of the main drawbacks of BERT is the **computational resources** needed to train/fine-tune and make inferences. In our task, we can see that NB performs much better (93% accuracy) than BERT (89% accuracy) on original data, and NB also requires much less time than BERT (seconds versus an hour). This is likely because BERT was designed to perform sentence-level tasks that require an understanding of the relationship between words and sentences. However, for our task, we can easily classify these texts

only by looking at separate words in the text without having to know the whole context. This is probably the reason that the NB model performs better on our task as compared to BERT.

VI. Conclusion

In summary, both the Multinomial Native Bayes and the BERT model performed reasonably well in classifying the industry of a business. If trained on a sizable dataset, the language models can most certainly provide a solid foundation for myriad use cases, including generating company descriptions based on the type of business and evaluating the communication effectiveness of any existing company profiles. We believe that such applications of language models help tailor company overviews to specific industries and thus allow organizations better communicate with their target audience. For future research, researchers may want to include more industry classes to enable broader application.

VII. Reference

Devlin, J. *et al.* (2019) *Bert: Pre-training of deep bidirectional Transformers for language understanding*, *arXiv.org*. Available at: https://arxiv.org/abs/1810.04805 (Accessed: December 16, 2022).

Harris, C.R. *et al.* (2020) *Array programming with NumPy*, *Nature News*. Nature Publishing Group. Available at: https://www.nature.com/articles/s41586-020-2649-2 (Accessed: December 18, 2022).

Pedregosa, F. *et al.* (1970) *Scikit-Learn: Machine learning in Python*, *Journal of Machine Learning Research*. Available at: http://jmlr.org/papers/v12/pedregosa11a.html (Accessed: December 16, 2022).

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). *Language models are unsupervised multitask learners*. OpenAI blog, 1(8), 9.(Accessed: December 19, 2022).

Usuga-Cadavid, J. P., Lamouri, S., Grabot, B., & Fortin, A. (2021). *Using deep learning to value free-form text data for predictive maintenance*. International Journal of Production Research, 1-28.(Accessed: December 19, 2022).