Project Report
Yuning Shen
CMPSC 273

Mining Microbial Interactions from Paper Abstracts using Text Classification

1 Background

With the increasing awareness of significance and complexity of microbial community, microbial consortia has become one of the most popular area in modern microbiology research. One of the key steps to understand the dynamics and functions of microbial consortia is to identify the interactions (or possible interactions) between microbes (e.g. bacterium A inhibits/benefits bacterium B). Despite its great importance and numerous novel experimental tools including DNA sequencing developed in recent years, inference of interactions still suffers from many statistical difficulties, for example, spurious interaction estimation from composition data and the sparsity of samples compared to the diversity of species.[1] At the same time, many experimental methods are still serve as the "golden standard" for microbiologists to assess the microbial interactions. However, this kind of results is scattered around numerous research papers on internet and it is not practical for any researcher to fully read and extract information from this large amount of literature.

Meanwhile in the discipline of natural language processing and knowledge base, extracting and summarizing information from a large amount of text data has been extensively studied and applied to many neighbor domains.[2, 3, 4] Thus, one possible solution is to build a system that can automatically scan and identify interactions from published research paper, and further construct, update and maintain a knowledge base of microbial interactions for researchers to validate, query, or integrate to their own analysis pipelines. As one of the fundamental tasks to achieve this goal, classify research papers into suggesting interactions between microbes can not only serve as a pre-filter before manual annotation of interactions, but also a valid method to gain statistical insight of interactions between two species.[1, 5]

Even though the great importance of the topic to microbiologists, as well as the development of text-mining techniques, there is very limited effort has been done on mining microbial interactions information from literature. To our knowledge, @MInter is the only machine learning model that to classify the abstracts into the class of suggesting the interaction of microbes and not suggesting the interaction of microbes.[5] In this project, we expect to build on this published paper and apply the recently developed transfer learning method BERT (Bidirectional Encoder Representations from Transformer) to the task of classification[6].

2 Task and methods

2.1 Task statement

In this course project, we will work on a binary classification problem that to classify input text strings (paper abstracts) into positive abstracts suggesting the interactions among microbes and negative abstracts not suggesting the interactions among microbes. It is a supervised problem and we will utilize the manually annotated abstracts from @MInter (3,917 annotated abstracts with 241 positive abstracts, as reported) to train classification models. Here are examples of positive and negative abstracts from the dataset:

- Positive

  Three peptides produced by a *Lactobacillus acidophilus* DPC6026 fermentation of sodium caseinate and showing antibacterial activity against pathogenic strains *Enterobacter* sakazakii ATCC 12868 and *Escherichia coli* DPC5063 were characterized... These peptides may have bioprotective applicability and potential use in milk-based formula, which has been linked to *E. sakazakii* infection in neonates.

- Negative

  ... *C. perfringens* and *B. fragilis* were discovered to have the strongest ability of degrading quercetin. Additionally, quercetin can't inhibit the growth of C. perfringens. In conclusion, many species of gut microbiota can degrade quercetin, but their ability are different.

In the above examples, orange words are the microbial entities, magenta words are the words suggesting the interactions between microbes, and the green words that are words also suggesting the concept of "interactions" but not among microbes. To classify the texts as showed above, we need to identify the keywords with the concept of "interaction" but also distinguish those false positives based on the context of the words.

We will use two models to achieve this task: one is model suggested in @MInter that used bag-of-word to embedded the text data, TF-IDF to account for the importance of each word (term), and Support Vector Machine as the classifier; the other is to use BERT to embed the input text and use a single layer of fully connected neural network as classifier, see Models for detail.

In the project, our goals are expected to achieve in three folds:

- Learn and practice the implementation of machine learning models, specified for text data.
- Validate the performance of model proposed in @MInter by re-implementing the model.
- Build a BERT-based text classification model and tune the model to achieve higher performance.

2.2 Models

2.2.1 BoW+SVM from @MInter

@MInter uses BoW (bag-of-word) to embed text (sentence, paragraph or document) into a feature space. BoW is a naive text embedding method by simply using the count for each word as a feature, considering neither the grammatical and semantic structure of original text nor similarity between words.

TF-IDF (Term Frequency-Inverse Document Frequency) is a heuristic method accounts for the "importance" of the word feature.[7] It has two parts that TF part discounting the multiple occurrence of words in longer documents and IDF part discounting the popular words in different documents.

SVM (Support Vector Machine) is a supervised machine learning model that by principle to find the best separator between two labeled classes by maximized the margin in between classes. It has been widely used in a broad range of domain including text classification.

In this model, each abstract will be embedded using BoW method and the value of each feature will be adjusted by TF-IDF before feed into the SVM for the training of classification. All three tasks has pre-build models in scikit-learn package, and in this project we will use scikit-learn for the implementation of this model.[8]

2.2.2 BERT-based classifier

In this project, we expected to use BERT to encode the entire abstract into a feature space (768 dimension from [CLS] by the design of BERT) that preserving the grammatical an semantic information of the original text, appended with a simple single softmax layer of neurons as classifier (Figure 1). BERT is one of the most advanced model built on the encoder structure of transformer. It has been used in many natural language processing tasks and outperform many existing state-of-art methods since publication. There are several features from BERT that might benefit our task:

- BERT preserves the position information of tokens by encoding the positional information into the input tokens and considering the complex dependencies among words through a multi-layer multi-head attentions mechanism.
- BERT model gained its great power from transfer learning. It is pre-trained with a large corpus (Wikipedia and BookCorpus) on two general language tasks (masked word prediction and next sentence prediction) with great computation power and time. The pre-train the model can be further trained (fine-tuning) for other language related tasks (text classification). Benefited from pretraining, fine-tuning a model that can usually get better results with faster convergence and on a traditionally small dataset (our task), compared to train the task from the scratch.
- Another embedded mechanism used in BERT is the tokenization model WordPiece that can resolve the problem of out-of-vocabulary word (OOV) by dissecting the unseen word into ”pieces” that are contained in the vocabulary. This feature is important for our text which contains an above average amount of rare words that are only used in scientific domains.

Among different versions of pretrained BERT model provided by Google, we will use BERT-Base model (uncased) version, in consideration of our computation resource and time. In addition to the pretrained BERT model from Google for general purpose of language modeling, we also used BioBERT, a same BERT-Base model that continued trained on biomedical domain specific corpus (PubMed and PMC) as our base model for fine-tuning.[9] BioBERT is preferred as pretraining was conducted on biomedical dataset. Reported results in this reported are based on the fine-tuning on BioBERT, if not indicated specifically. BERT model is implemented and modified using tensorflow package in python.

All the experiments were conducted on a local linux machine with Intel Xeon E3-1200 Processor and NVIDIA GeForce GTX 1080 graphic processor with 8 Gb memory. Project code is deposited at https://github.com/ynshen/cs-273

3 Experiments and results

3.1 Abstract data preprocessing

To prepare the labeled abstracts for our proposed models, we first combined and converted the @MInter-specific annotated data files (.ann) into a .tsv file with two columns (label and abstract).

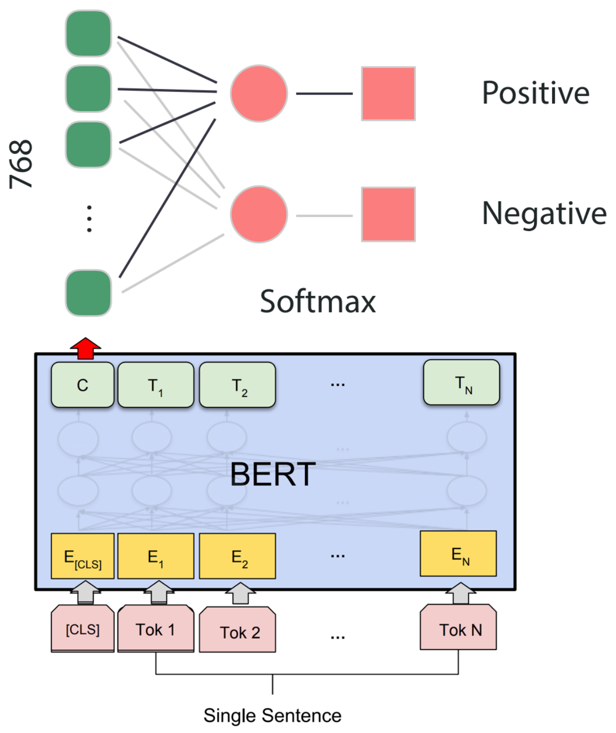


Figure 1: Structure of proposed BERT-based classification model.[CLS] is the sentence level token used for text-level classification, it has output with 768 dimension can passed to downstream classifier.

We inspected the length distribution of the abstract text by splitting the abstract string with respect to space ' ', and find that most of abstracts are within length of 150 to 300 words, as showed in Figure 2. In addition, we found around 190 negative abstracts included in the annotated dataset do not contain valid abstract (with only a placeholder token). After removing invalid abstracts, we obtains a dataset contains 3,954 abstracts with 294 positive abstracts and 3,660 negative abstracts.

We used two different approaches to split dataset for training and evaluation: 1) classic training-validation-test split with ratio of 0.8, 0.1, 0.1 for preliminary investigation and 2) 5-fold cross validation for the actual training. In these two approaches, training set size is designed to be same (80 % of entire data) and in data splitting, we used stratified splitting to ensure that each set has similar number of positive samples and negative samples. Cross validation method is preferred and used as major training scheme given our relative small dataset and the concern of variation among data splits.

3.2 BoW-SVM Model from @MInter

We first implement the proposed BoW-SVM model in sci-kit learn to re-evaluate the model from @MInter as baseline. The Bag-of-word construct and TF-IDF normalization was implemented using integrated function in sci-kit learn. As suggested in the paper, we used 'linear kernel' with default penalty parameter (C=1). Training dataset is balanced by the inverse of occurrence of data type (Pos/Neg) using embedded argument in sci-kit learn SVM. The model was evaluated by 5-fold cross validation and the accuracy, precision, recall and F1 score, along with their mean and standard

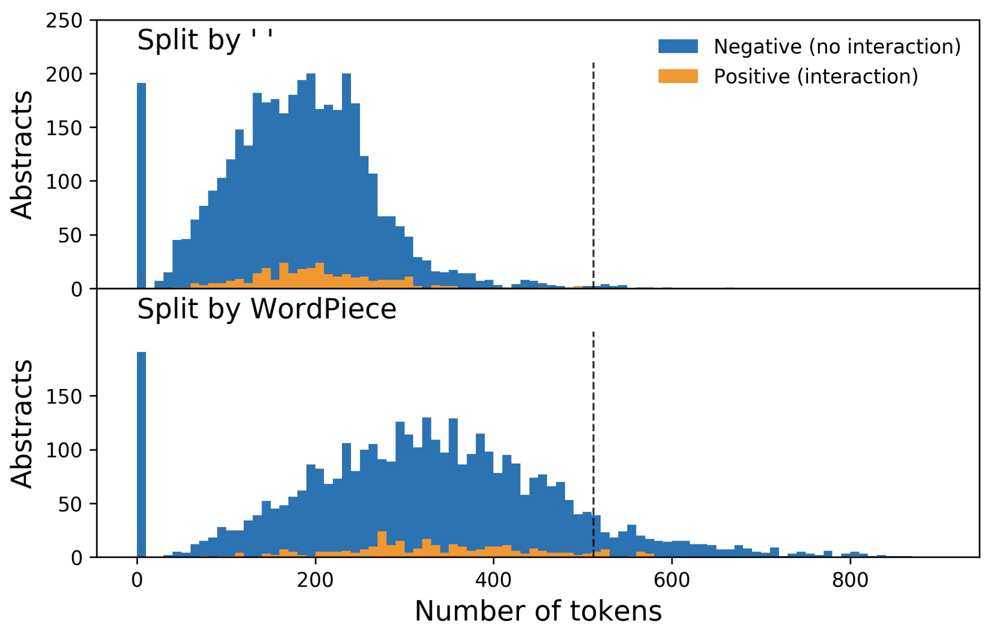


Figure 2: Length distribution of tokenized abstract data generated from splitting by space ' ' (top); splitting by WordPiece embedded in BERT (bottom). Vertical dash line indicate length 512, which is the maximal input length of BERT model.

deviation are calculated in Table 1. We compared these scores to the reported scores (with 0.5 as cut-off threshold for classification) in the paper. The accuracy and F1 score are not reported in original @MInter paper but could be calculated from reported Precision and Recall with given sample size.

	Our implementation	**Reported**
Accuracy	0.959 ± 0.011	0.96
Precision	0.717 ± 0.077	0.49
Recall	0.748 ± 0.059	0.82
F1	0.732 ± 0.063	0.61

Table 1: Results from BoW + SVM

Surprisingly, BoW-SVM model we implemented performs significantly better than the reported values in terms of precision and F1 scores; however, slightly lower in recall score. This indicates the implemented BoW-SVM model is more capable of distinguishing false positives from true positives and with slight scarification on identifying true positives. And the performance is consistent in 5-fold cross validation (Figure S2). We noticed that there are two differences in the implementation could possibly contribute to the difference of results: 1) the actual dataset size is different from the dataset size reported in the paper and we filtered out some invalid abstracts; 2) in the original paper, authors always includes a ”core dataset” that only includes abstract from the search of two bacteria species (*Lactobacillus acidophilus* and *Esherichia coli*) in the all training sets and we treat all abstracts equally in our implementation.

3.3 BERT Model

We next move to proposed BERT Model to classify abstracts. There are several challenges during the implementation of BERT model:

- As indicated above, original BERT model can only encode the sequence of tokens with maximal length 512, including the [CLS] token necessary for classification. However, as showed in Figure 2, using the WordPiece tokenizer will extend length of tokenized abstract above the limit, because of the word splitting feature and many out-of-vocabulary words in scientific abstracts. We used a heuristic method that truncate the abstracts that are too long from the top, based on the knowledge that many long abstracts start with sentences introducing the research background and irrelevant to the suggestion of interaction.
- In order to accommodate the long input sequence as well as the limitation of computation power, we compromised by using a smaller batch size for training to avoid memory allocation error. We found the maximal available batch size for training allowed on our 8G machine is 4 and we fixed it for all the experiments.

3.3.1 Monitor the training curve

In order to find the necessary training epochs for BERT model, we modified the code in the original BERT classifier to evaluate and record relevant metrics on both training and validation set (or test set in cross validation) during the training process. We first tested on the train-validation-test split data and concluded that the optimization of loss stopped after 2,000 data batches (roughly 2.5 training epochs) for both fine-tuning on BioBERT (Figure 3) and BERT models (Figure S1). Over-fitting is observed for fine-tuning on BioBERT models, led to shortened training epochs for cross validation analysis.

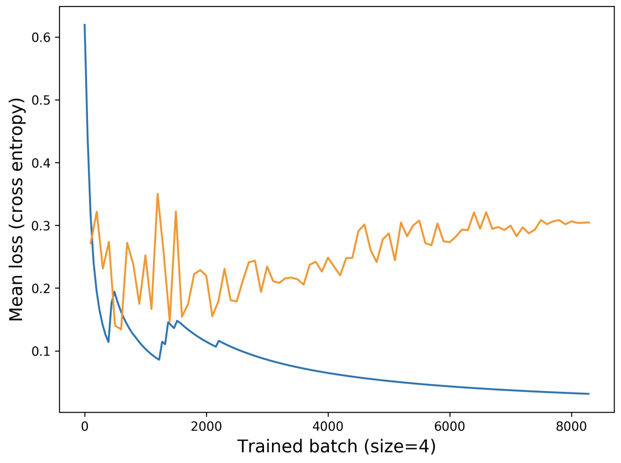


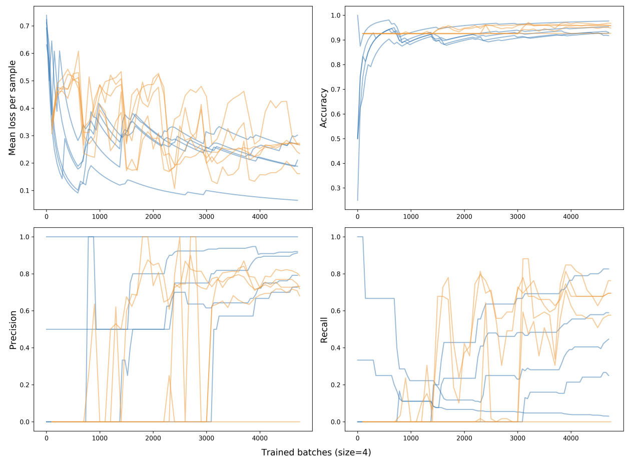
Figure 3: Training curves on BioBERT to find out optimal training epochs. (Blue: training set, Yellow: validation set)

3.3.2 Cross validation on BERT Models

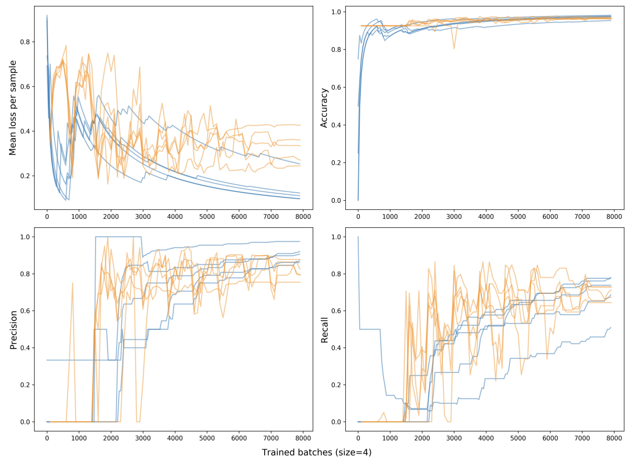
We further used cross validation to assess the robustness of the model training by fine-tuning on BioBERT model up to 6 epochs of data based on the observation from previous section. We recorded the change of relevant metrics during the training (Figure 4a).

Unexpectedly, unlike training curve obtained in train-validation-test split, we did not see a converged result of training in cross validation experiments. In the other word, the model is still "underfit" given the data and train epochs. For some typical sets (set 1, 4, 5 for BERT and set 5 for BioBERT), training is barely started within in 6 epochs, leaving the precision and recall value to be close to zero that almost all the positive text are misclassified as negatives. A further inspection from comparing curves of precision/recall to accuracy indicated that the inefficiency of training might be not only from the too short training time but little contribution from positive samples. As they are under-represented in the training data, it seems the training process is mostly focused on optimizing the contribution from probability of already correctly classified negative samples instead of correction of misclassified positives samples.

To account for these problems, we decided to extend the training epochs and use weighted loss function for the training to account for the imbalance of data.



(a) Unweighted loss function, 6 epochs



(b) Weighted loss function, 8 epochs

Figure 4: Weighting loss function improve the convergence of training in 5-fold cross validation experiments on BioBERT models. (Blue: training set; yellow: test set.)

3.3.3 Weighted BERT Model

We adjusted the loss function of model by weighting the contribution of positive and negative data by the inverse of their occurrences, specifically:

$$L = \frac{1}{n}\sum_{i=1}^{n}(-w_0 y_i \log(p_{1,i}) - w_1(1-y_i)\log(p_{0,i}))$$

$$w_0 = \frac{N_{\text{Neg}} + N_{\text{Pos}}}{N_{\text{Neg}}} \quad w_1 = \frac{N_{\text{Neg}} + N_{\text{Pos}}}{N_{\text{Pos}}}$$

where n is the batch size ($n = 4$), y_i is the label for i-th data, $p_{1,i}$ and $p_{0,i}$ are the predicted probability to be positive or negative; N_{Neg} and N_{Pos} is the total number of negative abstracts and positive abstracts in entire dataset. We did not weight the occurrence of positive and negative samples per batch base due to the small batch size we used.

We can see from the training curve of model with weighted loss function as well as extended training time effectively helped the training process and all the models seem consistently converged towards the end of training period, by observing precision and recall. We also did another experiment with 10 epochs of training and achieved same results (Figure S4)

To pick the best model from the training, we selected the model in the converged region with minimal mean loss. We picked the best model for each set of training in cross validation and calculated the mean and standard deviation of different metrics. The best models decided from 8 epochs training and 10 epochs training are compared to our implementation of BoW+SVM model and best performer is highlighted in bold for each metric, as showed in Table 2.

	Accuracy	**Precision**	**Recall**	**F1**
BoW+SVM	0.959 ± 0.011	0.717 ± 0.077	$\mathbf{0.748} \pm 0.059$	0.732 ± 0.063
weighted BioBERT (8 epchs)	0.962 ± 0.009	0.751 ± 0.059	0.724 ± 0.137	0.825 ± 0.094
weighted BioBERT (10 epchs)	$\mathbf{0.964} \pm 0.010$	$\mathbf{0.778} \pm 0.045$	0.728 ± 0.120	$\mathbf{0.829} \pm 0.079$

Table 2: Training results on best models

Weighted BERT model fine-tuned from BioBERT pretrained model out-performs BoW+SVM we implemented, even though the later one already has significant improvement comparing to the reported performance, see Discussion for detail.

4 Discussion and conclusion

4.1 Model training results

In this project, we successfully implemented BoW+SVM model using sci-kit learn and weighted BERT in tensorflow. As summarized in Table 2, best models selected from weighted BioBERT trained with different epochs have comparable results and both out-perform BoW+SVM model in the most relevant metrics. Specifically, BERT model further improve the precision which indicates the percentage of true positives over the total selected positives (both models has higher precision comparing to the reported value). In our case, model with high precision can best serve the purpose that to pre-filter the abstracts that might suggest the interactions between microbes to decrease the labor of manual labeling. In order to further evaluate the performance of models serving for different purposes (e.g. high recall for detect newly published papers regarding microbial interaction), we need to plot ROC (Receiver Operating Characteristic) curve and Precision-Recall curve to see the trade off of different metrics with respect to different level of probability thresholds in classification.

4.2 Variance in the results

We do notice that the variance of each metric is also large. To draw a more statistical reliable conclusion, we need to conduct more repeated experiments (e.g. 10 repeats) to assess the stochasticity among the model training and experiment on different splits of the dataset to account for the variance in the data. However, a too small dataset might just be not representative enough and it is not easy to avoid the fluctuation among different subset of data. One solution is to find some closely relevant tasks (e.g. mining interaction/association/relation in other scientific domains) to enrich the dataset for training (or pretraining). Another solution is to build an efficient crowd-sourcing platform to allow microbiologists or educated public to annotate the data manually, or set up a automatic feedback system to validate the predicted information from authors.

4.3 BERT model improvement

Due to the time and resource constrains on this project, we were not able to experiment or extend the BERT model in many aspects. For example, we would love to try different learning rates or batch sizes during the training and to see how to further optimize the training process. Also, we could apply other classifiers after BERT model instead of a single softmax layer for classification - from current observation, some positive data are not able to distinguish from negative data. This could possibly come from the naive classifier we chose and the basic version of BERT model we implemented.

4.4 Other tasks and future directions

Even though our BERT-based model has not showed an absolute win over the previous model (BoW + SVM) in the current implementation, BERT was chosen also because its versatility to be used for related tasks. As stated in the introduction, current project is a simplified version of microbial interaction extraction to construct a microbial interaction network. Instead of ”sentence level” embedding from [CLS] token, BERT could generate token level embedding for the original text, which is possible to be implemented in the microbial name entity recognition and identify interactions between typical bacterial entities.

References

[1] Chieh Lo and Radu Marculescu. “MPLasso: Inferring microbial association networks using prior microbial knowledge”. In: *PLoS computational biology* 13.12 (Dec. 2017), e1005915.

[2] Graciela H Gonzalez et al. “Recent Advances and Emerging Applications in Text and Data Mining for Biomedical Discovery.” In: *Briefings in Bioinformatics* 17.1 (Jan. 2016), pp. 33–42.

[3] Martin Krallinger et al. “Information Retrieval and Text Mining Technologies for Chemistry”. In: *Chemical Reviews* 117.12 (June 2017), pp. 7673–7761.

[4] Xiaoyan Wang et al. “Bacterial named entity recognition based on dictionary and conditional random field”. In: *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, pp. 439–444.

[5] Kun Ming Kenneth Lim et al. “@MInter: automated text-mining of microbial interactions”. In: *Bioinformatics* 32.19 (Oct. 2016), pp. 2981–2987.

[6] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *arXiv.org* (Oct. 2018). arXiv: 1810.04805v2 [cs.CL].

[7] J Ramos Proceedings of the first instructional conference on and 2003. “Using tf-idf to determine word relevance in document queries”. In: *researchgate.net* ().

[8] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[9] Jinhyuk Lee et al. *BioBERT: a pre-trained biomedical language representation model for biomedical text mining*. 2019. arXiv: 1901.08746 [cs.CL].

5 Supplementary information

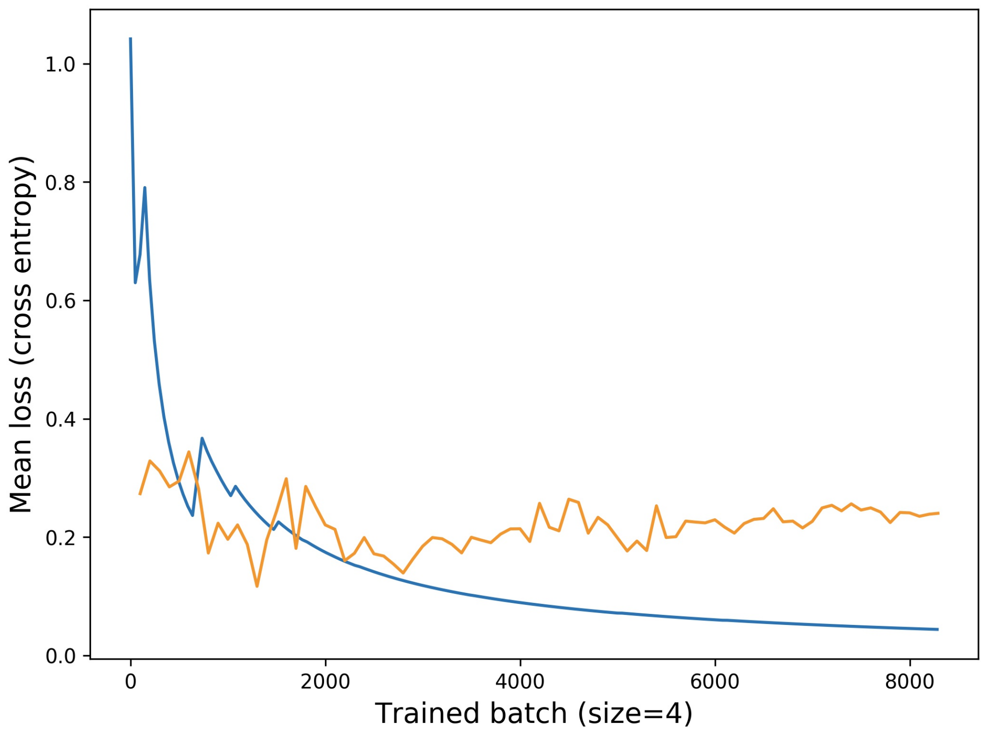
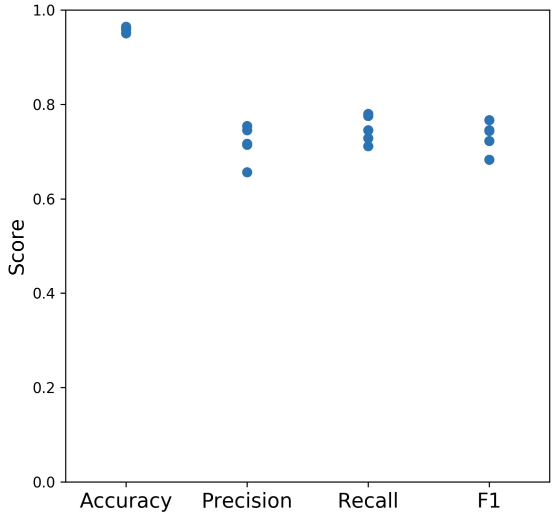
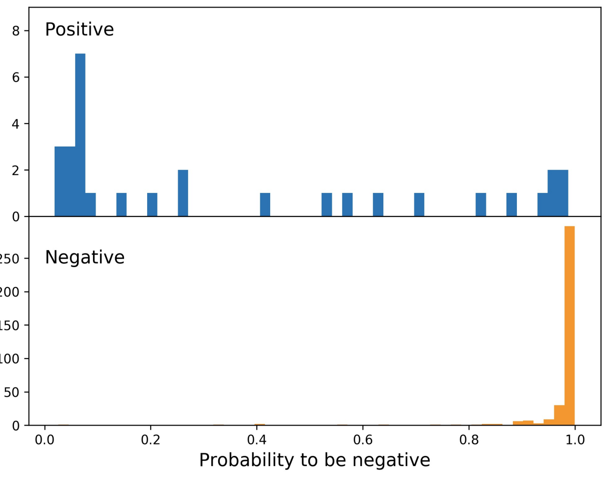


Figure S1: Training curve on mean loss from fine-tuning BERT



(a) Scores



(b) Distribution of probability of test set

Figure S2: Results from 5-fold validation of BoW-SVM Model

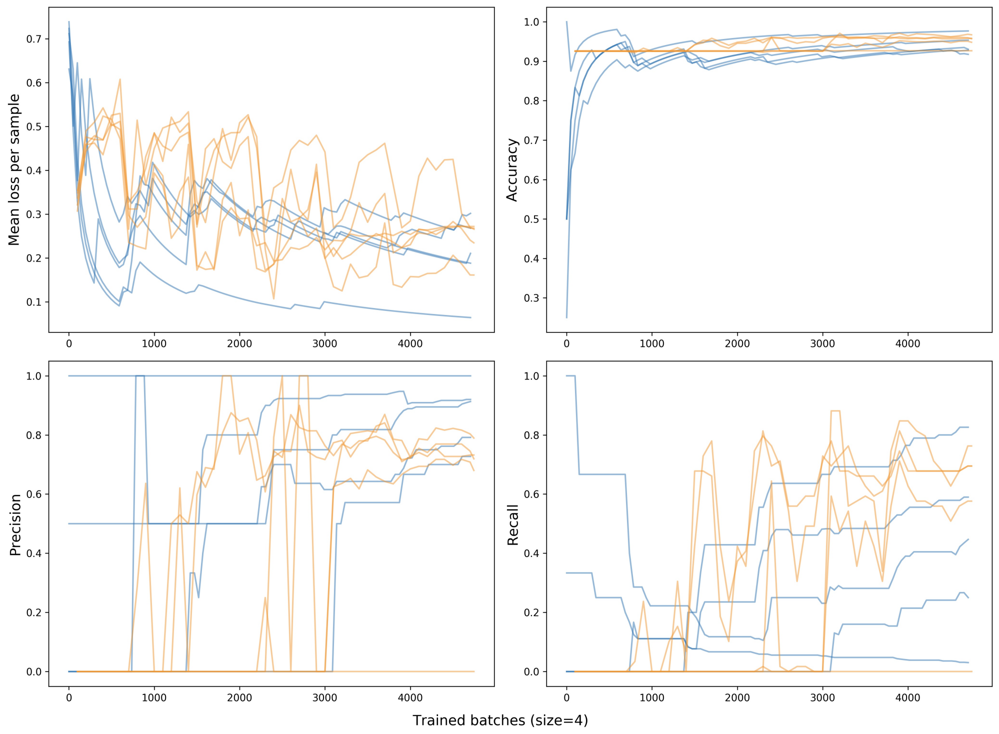


Figure S3: Learning curves on unweighted model fine-tuning on BioBERT with 6 epochs. (Yellow: test set; Blue: training set)

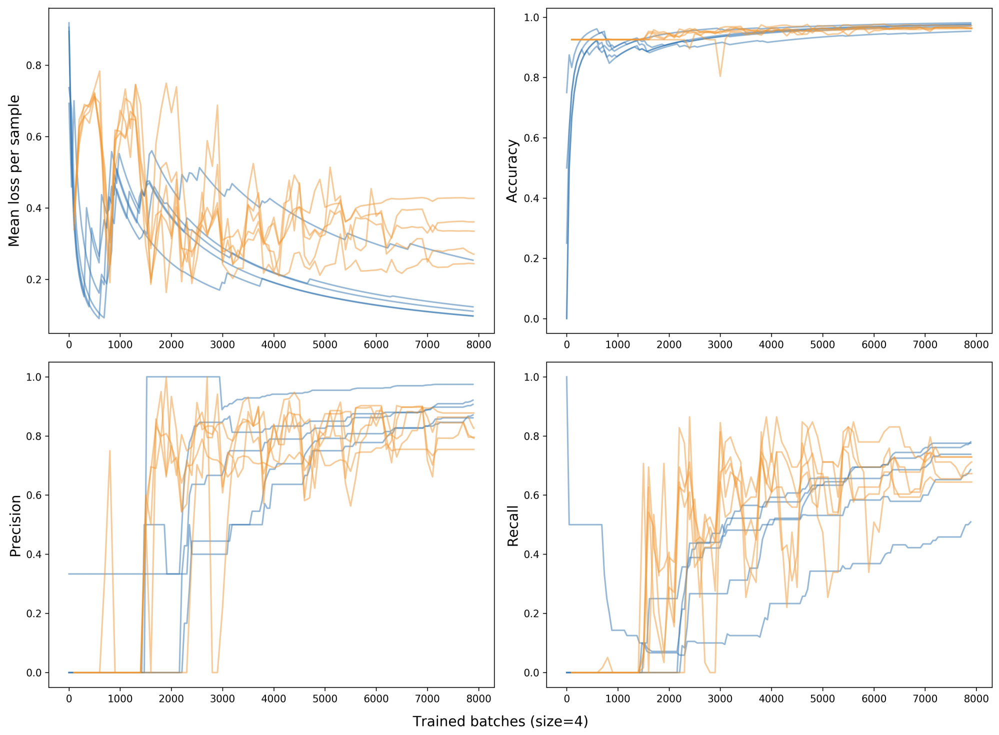


Figure S4: Learning curves on weighted model fine-tuning on BioBERT with 10 epochs. (Yellow: test set; Blue: training set)

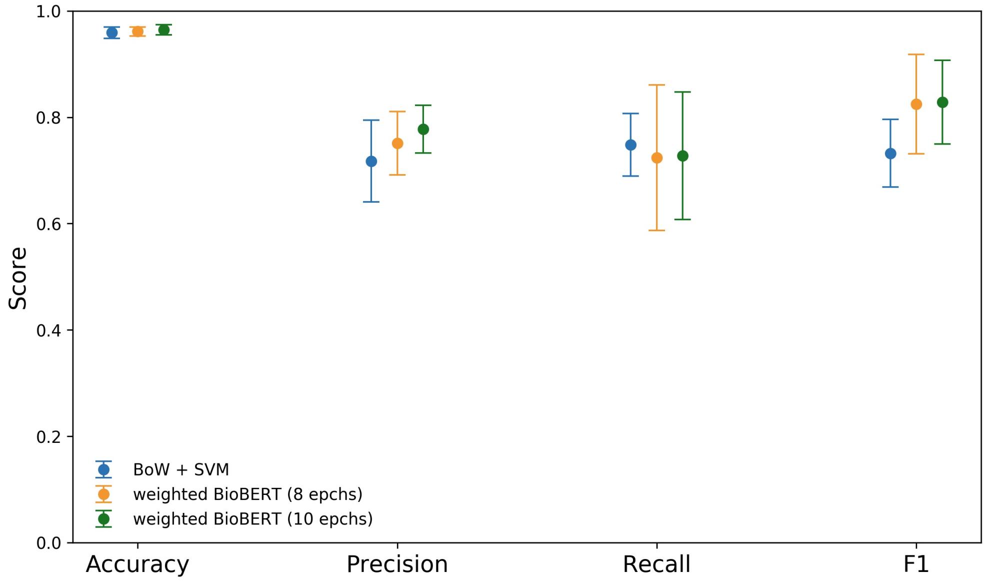


Figure S5: Learning results comparison among selected three selected models