Tampere University

Weerin Wongwarawipatr

BIOBERT FOR DIETARY COMPOUNDS AND CANCER RELATION EXTRACTION

Master of Science Thesis
Faculty of Information and Communication Technology
Examiners: Professor Frank Emmert-Streib, Professor Jaakko Peltonen
November 2021

ABSTRACT

Weerin Wongwarawipatr: BioBERT for Dietary Compounds and Cancer Relation Extraction
Master of Science Thesis
Tampere University
Master of Science in Computing Science - Data Science
November 2021

The relation extraction of biomedical publications has been an essential natural language processing task for constructing the biomedical network. This network will allow the study of genes, disease, and food compounds holistically together, which will help to solve many biomedical problems. Biomedical papers, which are growing exponentially over time, are the primary resource for network construction. Thus, natural language processing is an integral part of enabling this network creation from biomedical text papers. The fully connected biomedical knowledge graph that connects many sub-fields will benefit the better data management and analysis of text data in the biomedical domain.

While there are several pieces of research on genes and disease natural language text relation extraction, there are not many works on disease and food compounds conducted. This research aims to conduct the natural language relation extraction models for dietary compounds and cancer.

This research uses the transfer learning method to acquire the pre-trained models and fine-tune them with the prepared biomedical text dataset. The state-of-art natural language understanding model, BERT, will be explored with other BERT-based variations, DistilBERT and BioBERT. The fine-tuned BioBERT model is expected to give the best result in the end since it is the specialized model pre-trained with biomedical text documents. In addition, to benchmark the model performance effectively, we include the two traditional machine learning classification models, Support Vector Machine and Gaussian Naive Bayes, to be the baseline for the comparison with the proven state-of-art language models. The text data is acquired from PubMed search engine, and the articles are biomedical paper abstracts about cancer and food compounds. The food and cancer entities are annotated manually.

The result shows that BioBERT has the best performance with a 0.7855 F1-score. It is higher than the original BERT model for 0.0011 F1-score. DistilBERT also acquires acceptable performance with a 0.7338 F1-score. DistilBERT performs justifiable prediction with only 0.05 lower than BERT original model but 50% faster in model fine-tuning compared to its model size. We have also constructed learning curves to visualize each model learning process. All BERT-based models can be improved with more datasets in future research.

Keywords: Biomedical Text Mining, Relation Extraction, Natural Language Processing, Natural Language Understanding, Information Extraction, Transformers, BERT, BioBERT

PREFACE

I started my thesis with an interest in the natural language processing, which is the area I have explored during my bachelor's thesis in 2017. I was unsure of what application and area this topic could be applied to until I met professor Frank Emmert-Streib who have guided me through the biomedical research area. It is a new interesting area of research where I had never thought about, the demand for natural language processing in life science. I got a chance to hand on the state-of-art language model, BERT, where I find it fascinating and will be the skill that I can certainly apply in my future works or personal projects in the near future.

I would like to show my gratitude to Professor Frank, who guided and directed me to this demanding research area and connected me with many other talented researchers, Nadeesha Perera and Shailesh Tripathy. They both helped consult me on reference papers and the implementation using the huggingface library. Thanks for bearing with me answering my questions, and explaining to me the different topics that I didn't understand.

Finally, I would like to give thanks to my aunt, Tipaporn Amnatsatit, who is always my big supporter. She is a fantastic lady who raised me up alone on her own. Her vision of exploring the world getting experience outside of our homeland always pushes me forward and leads me to come study abroad for a Master's Degree at Tampere University. I am always grateful for you.

Tampere, 14th November 2021

Weerin Wongwarawipatr

CONTENTS

1 INTRODUCTION

Genetics, diseases, and food science are linked to each other according to the nature of substances. However, a holistic approach to study the documents from these three research areas entirely together has not yet been fully available. There has been an effort from the computer science community to create a biological network to study about genes and disease relationship (Emmert-Streib, Dehmer and Haibe-Kains 2014). The work by Emmert-Streib, Dehmer and Haibe-Kains 2014 has expressed the importance of gene regulatory network that it can help in solving many biomedical problems and to make the network completed, it should be linked with other life science networks to get the fully connected information. Due to the number of biomedical publications grows exponentially over time, text data in biomedical field can be the main resource for the network construction. The natural language processing is an important part to enable this network to be constructed from the text data. The fully connected biomedical network will benefit the better data management and the analysis of text data in the biomedical domain.

Natural language processing can enable the information extraction system from biomedical publications. Relation extraction is one of the important task that has an important role in linking the entities. It is the task to classify relation types of two or more entities appearing in natural language text. Many researchers conduct this task with biomedical entities, proteins, drugs, genes, and diseases (Perera et al. 2020). However, there have not been many pieces of research on relation extraction of dietary compounds associated with diseases conducted even though many papers about phytochemicals, nutrition, and food components related to diseases are available in biomedical databases. Hence, this research will perform the relation extraction between dietary compounds and disease, specifically cancer, from the biomedical papers' abstracts available on PubMed, a free search engine of MEDLINE database containing science journals about life science biomedical topics.

There have been many approaches to achieve the relation extraction task. They can be classified into four groups, co-occurrence-based, pattern-based, rule-based, and machine learning-based (Abdul Wahab Muzaffar 2015). Before processing relation extraction, the implementation of named-entities recognition to annotate the entities will need to be done. When the named entities have already been annotated, the machine learning process afterward will become more powerful (Giuliano et al. 2006).

This research will compare the result of relation extraction done by the natural language processing models, BERT (Devlin et al. 2019), DistilBERT(Sanh et al. 2019), BioBERT (Lee et al. 2019), comparing with the method done by traditional machine learning, Support Vector Machine, and Gaussian Naive Bayes Classifiers. The text data will be the biomedical papers on the diet
(food, phytochemicals, nutrition) associated with cancer, available on Pubmed, the biomedical papers search engine. Only the sentences in the abstracts part will be used as the dataset. In the end, the BioBERT model is expected to give the best result since it is the model trained with Biomedical text data, which should manifest specialty for natural text in biomedical field.

2 LITERATURE REVIEW

Information Extraction(IE) is the field of study for extracting information from natural language text. IE aims to create the understandable text for machines to process. It can be used for many tasks such as text summarization, question answering system, search optimization and etc. To extract information within the text, the first task to be done is Named-Entity Recognition(NER). NER automatically detects and locates keywords in a sentence that are classified as predefined categories. After performing the NER process, the Relation Extraction(RE) task can be done to process the relations between the extracted entities within the sentence. Therefore, RE is the task focusing on extracting semantic relations between the extracted entities in the text. There are many prominent works in the RE performed by biomedical text data. They have been proposed with many different approaches, which can be classified into four main groups.

2.1 Co-occurrence Based

Percha et al. 2011 conducted a co-occurrence approach for the relation extraction task. This method uses the frequency of occurrence between entities in the target sentences to determine the probability of association.

2.2 Rule-Based

This method relies on semantic analysis using part-of-speech to detect relations. Fundel et al. 2006 proposed syntactic parse trees to break down sentences to find the relation between noun phrases and verbs. This approach requires a predefined list of verbs that can identify relations between noun phrases, for example, prevent, inhibit, prohibit, accelerate.

2.3 Conventional Machine Learning

Machine learning models applied to the relation extraction are mostly supervised learning models that require an annotated dataset with a predefined target class for the models to learn during the training process. H. . Yang et al. 2011 had proposed the machine learning model, support vector machine (SVM), to classify a polarity and evaluate the strength level of relations between food-disease entities. This model is a multiclass classifier separating the polarity between positive, negative, neutral, and irrelevant associations.

Another machine learning model was explored by Jensen et al. 2014. This work had proposed a Naive Bayes model for classifying food-disease associations. The TF-IDF feature extraction was applied to the dataset before inputting it into the training process.

2.4 Deep Learning

The deep learning models commonly used for relation extraction are the convolutional neural network (CNN), recurrent neural network (RNN), and the combination between CNN and RNN (Emmert-Streib, Z. Yang et al. 2020). The input features can be the sentence-level or word-level embedding vector representation and the position of entities that appeared within the sentences. One of the work on using CNN for biomedical relation extraction was proposed by Liu et al. 2016. CNN was used for extracting drug-drug interaction(DDI), obtaining an F-score of 69.75%. The result outperformed SVM, the most state-of-art DDI extraction machine learning model at that time, by 2.75%. However, CNN requires the input data length to be similar, so zeros padding needs to be done. To further research, Sahu and Anand 2018 did DDI extraction by long short term memory network(LSTM), which is an RNN based model. The model has a word and a position embedding as latent features extracted from sentences. In contrast to CNN, LSTM processes the input vectors sequentially and has no restriction in the length of each input. Bi-LSTM is also explored in this work. It is a two way LSTM going forward and backward concatenated together. This model got the better performance as it allows the network to extract the implicit features from both left-to-right and right-to-left sequences in the sentences.

In 2017, the Transformers model was introduced in the publication; Attention is all you need by Vaswani et al. It introduced the self-attention or intra-attention mechanism, which can encode the input parallelly. The model structure entirely consists of different attention cells in both encoding and decoding parts without using any convolutional neural network(CNN) and RNN, LSTM cells. This introduced a better approach with the improvement in time consumption and the better quality of language understanding. Afterward, BERT (Devlin et al. 2019) had been established. BERT structure is simply the layers of the encoder part of Transformers stacking together to get the deeper context for language understanding which is the origin for its name, “Bidirectional Encoder Representation from Transformers”.

Shi and Lin 2019 had applied BERT-based models to the RE task. BERT can provide state-of-art performance using a simple neural architecture without the lexical and syntactic features like part-of-speech tags and dependency tree.

BioBERT (Lee et al. 2019) is the BERT architecture model that has processed the pre-trained stage with a Biomedical domain corpus which consists of PubMed abstracts and PMC articles. When comparing to the pre-trained weights from BERT that was pre-trained by Wikipedia and BooksCorpus, BioBERT improves the performance of the biomedical text relation extraction task by 2.8% of F1 score.

3 BACKGROUND THEORIES

3.1 Deep Learning Methods in Natural Language Processing

3.1.1 Transfer Learning

Transfer Learning is the method of transferring the knowledge gained from the prior model training. The trained model weights will be reused and adjusted according to the dataset in another related task. The first stage of training the model weight from scratch is called pretraining, while the latter stage of training the trained model weights to fit in with a particular task is called fine-tuning.

Transfer learning has become a prevalent method for utilizing available pretrained models. During the pretraining process, the model requires much cost in computation and time. For the model solving Natural Language Processing tasks, Transformer architecture has become very well-known and has been used as the base model for other language models. It is shown that more parameters and more datasets during the training process improve the model performance. However, the larger models cost a lot in computing resources, time, and even environmental impact. It is shown in the work by Strubell et al. 2019 that training a Transformer model with 213 Million parameters can emit over 626,000 pounds of carbon footprint. Transfer Learning is the way to utilize the knowledge gained from pre-trained models. To fine-tune the trained model weights instead of training it from scratch will decrease the time, computational cost, and even the global CO2 emissions.

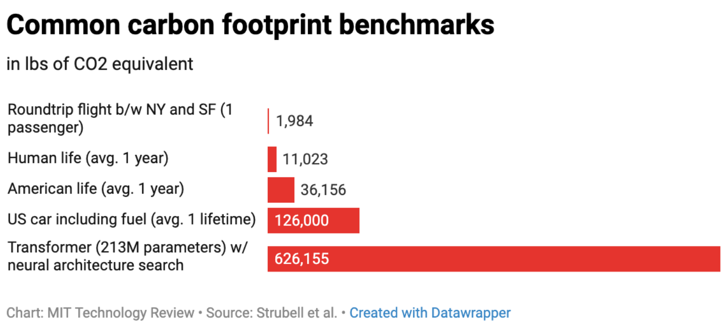


Figure 3.1. *Training a single AI model can emit as much carbon as five cars in their lifetimes (MIT Technology Review 2019).*

Hence, low-resource models become more demanding. One of the outlier models, DistilBERT, which will be explored in this research, only has 66 million parameters but still maintains a high performance comparable to other state-of-art models.

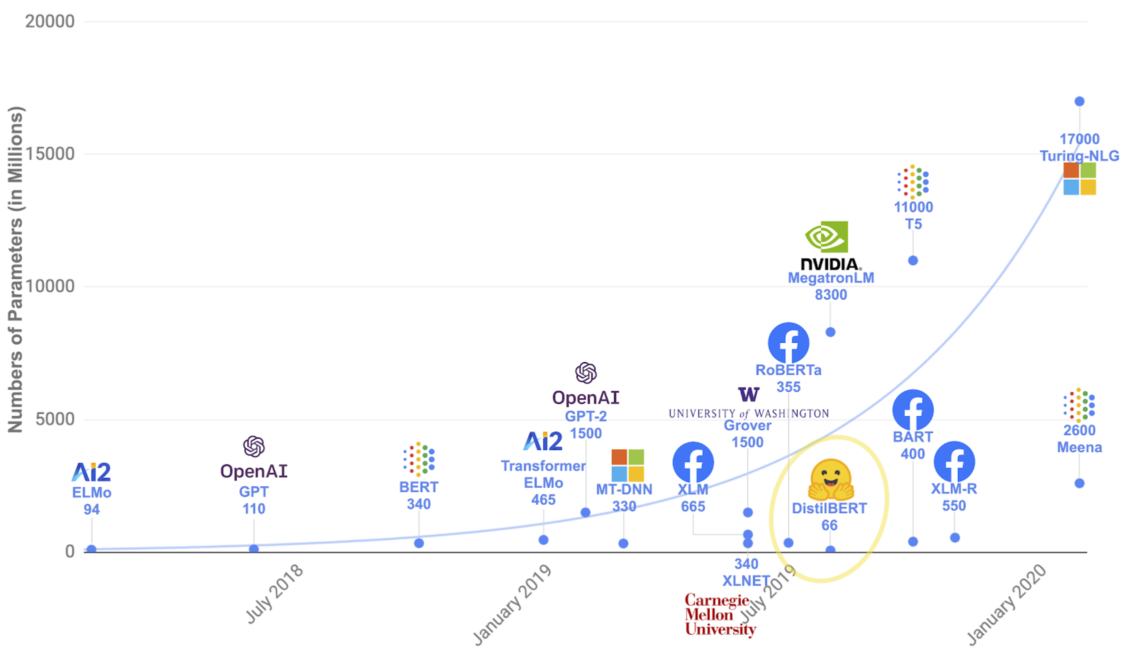


__Figure 3.2.__ The size of NLP models through time (TensorFlow Blog 2020).

3.1.2 Transformers

Transformers(Vaswani et al. 2017) was introduced in June 2017. It was originally made for translation tasks. The intuition was from improving an issue with translating the long sentences. Before it was introduced, Recurrence Neural Network (RNN) and Convolutional Neural Network (CNN) with an encoder and decoder were the most prominent solution for machine translation as it was designed to work with sequence-to-sequence model. However, when the input sentences get longer, they become less efficient and consume much time. The intuition concept of Transformers was from the natural way of humans translating text by paying attention to each word and its contextual relating words within the sentence. This approach is more effective than having the model use all the words in the sentence with equal weights during the computation process. The key feature of Transformers that makes it different from RNN is the ability to train all words in the sentence simultaneously, takes lesser time to train, and achieves the better performance. Transformers is the state-of-art model for transduction problems, such as language modeling and machine translation. Moreover, it has been used as the base model for many of other state-of-art language models, such as BERT, GPT, RoBERT ,and GPT-3.

We can categorize the transformer based models into 3 categories, autoregressive, autoencoding, and sequence-to-sequence models.

- Autoregressive Models / Decoder Models
  The models are based solely on decoder layers of transformer model stacking on each other. It predicts the future result based on only the past outputs. This type of model is designed for a text generation task. The Example of this type of models are GPT, GTP-2, GPT-3, and Transformer-XL.
- Autoencoding Models / Encoder Models
  The models only use the encoder part of the transformer model. It can access all the words in the sentence in each stage simultaneously. This feature has been called bi-directional attention. This type of model is good for natural language understanding tasks, such as words classification, relation extraction, and question answering tasks. The example encoder models are BERT, ALBERT, and RoBERTa.
- Sequence-to-Sequence Models
  The Transformer model is a sequence-to-sequence model where both the encoder and decoder parts have been used. It is suitable for the task that requires sentences as the inputs and also outputs the result as sentences. Hence, the machine translation task is the perfect case. Other examples of language models with this architecture are Pegasus, BART, and T5.

The Transformer Model Architecture

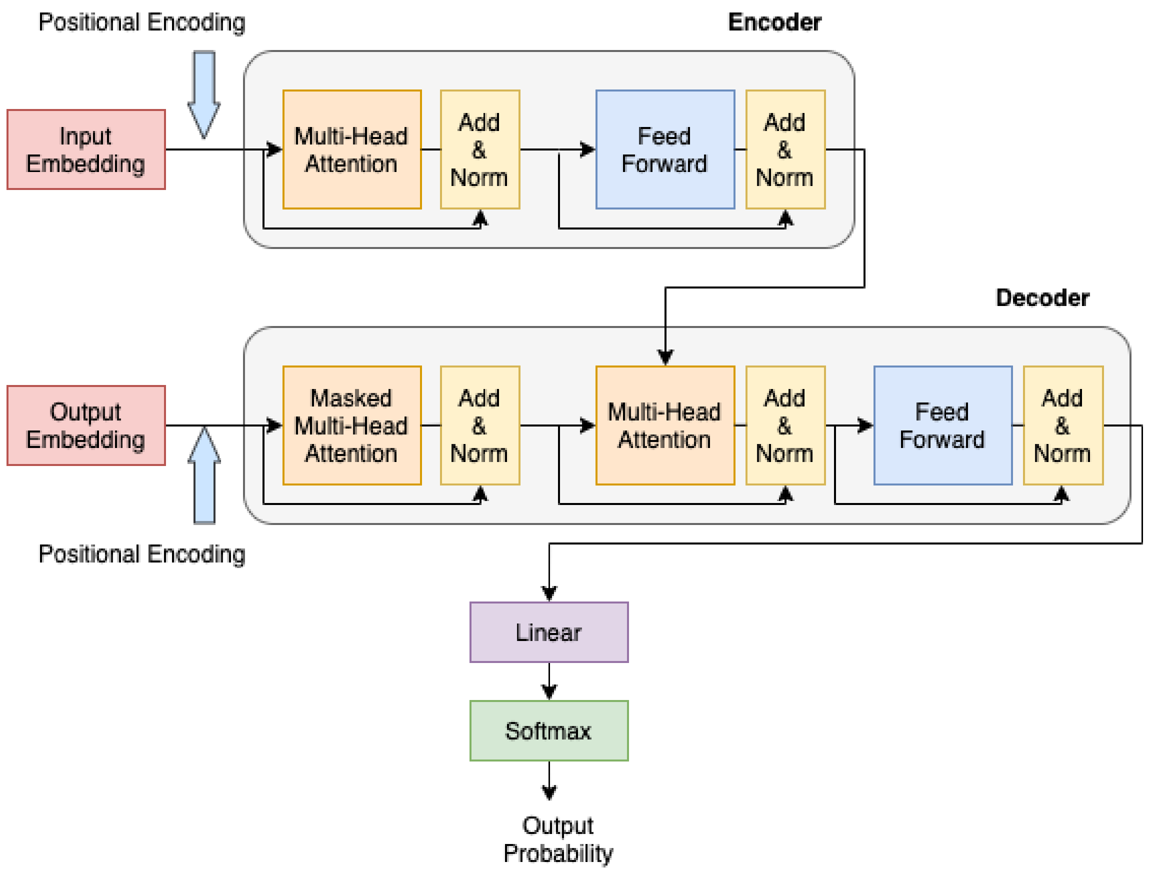


Figure 3.3. *The Transformer model architecture*

As can be seen from the figure 3.3, the model has mainly the encoder and decoder chunks. It does not incorporate CNN or RNN layers. Instead, it has the Multi-Head attention blocks as the primary critical components for contextual text processing, which we will describe further in the latter topic.

Encoder

The encoder consists of 2 sub-layers, a multi-head self-attention and a feed-forward neural network. In each sub-layer, the output is normalized by the result from the sub-layer added up by the input vector from the previous stage. This is to reduce the vanish gradient problem during the backpropagation process. The model consists of 6 encoder layers stacking on each other. It receives and produces the vector with 512 dimensions.

$$LayerNorm = (x + Sublayer(x)) \tag{3.1}$$

<u>Decoder</u>

The architecture within the decoder layers is overall the same as the encoder, except for the additional multi-head attention layer that receives the input from 6 encoder layers. All sub-layers are normalized with the input from each stage in the same fashion as in the encoder layer. Another prominent feature is the self-attention mechanism in the decoder layer. It prevents the computation of subsequent words in the sentence compared to the focus word in each iteration. It only computes attention for the words that position before the word embedding input in that stage. There are six decoder layers. All of them receive the input from the last encoder layer similarly.

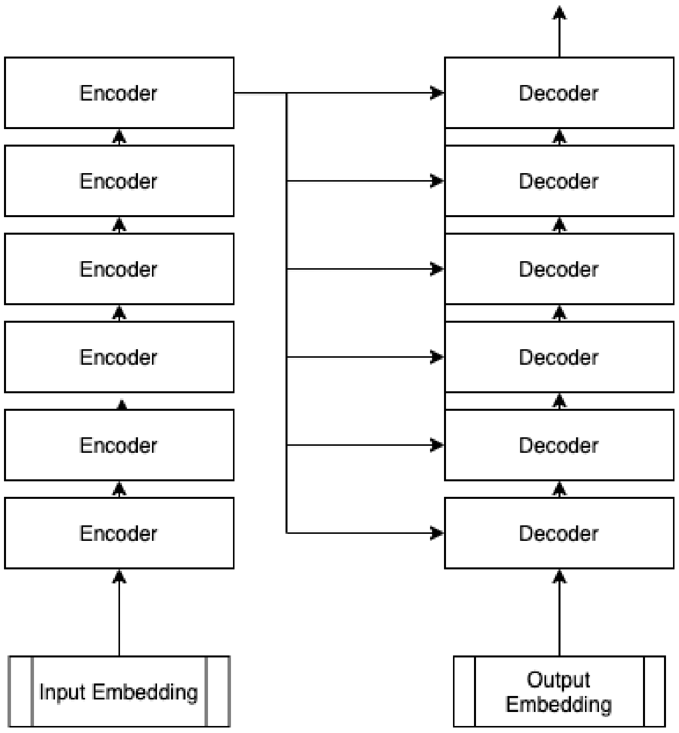


Figure 3.4. *The Illustration of how data pass between encoder and decoder*

Attention Mechanism

The attention mechanism is the key part of transformer architecture. It enhances the contextual meaning within text data. This valuable feature is acquired by the numerical operation of the input word embeddings. The input word embeddings are converted into three matirces, Query(Q), Key(K),and Value(V). These three matrics conversion is simply proceeded by multiplying the input word vectors with three matrices that have been initialized with some values and are trained during the training process.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{3.2}$$

$$Softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}} \tag{3.3}$$

We can describe the concept of the attention formula (Equation 3.2) in a simpler way by seeing the calculation from each row of the matrix one by one. In the vector level, the formula 3.2 is simply to acquire the dot product of the similarity between the word vector through other words in the sentence and its value vector (V_i). The similarity of each word embedding vectors $(Q_i K_{1\to n}^T)$ is the dot product of the query vector representing the specific word Q_i and key vectors of other words in the sentences including itself $K_{1\to n}$. According to the equation 3.2, the dot product of the query and the key vectors is divided by the square root of key vector dimension ($\sqrt{d_k}$). This is for decreasing the large magnitude value when applying the softmax function afterward. The softmax function is applied to scale the dot product for having them sum up to 1 and making it suitable for using as the weight for the attention value calculation.

The attention mechanism is applied to self-attention layers where all query vectors process sum-product operation with their own and other key vectors in the sentence. Moreover, it has also been used in the second Multi-Head Attention layer in the decoding block layers where it receives inputs from the encoder, as can be seen in the figure 3.3. This layer can be called the encoder-decoder layer. In this is attention layer, the input from the encoder is used as key vectors operating dot product with the query vectors from the output of the previous attention layer in the decoder stack itself.

Instead of finalizing the model parameter based on one attention, the transformer model uses the multi-head attention to attend more information from h attention heads. The multi-head attention is the method of computing h attention layers in parallel and concatenating the result into an output vector with the set dimension. This is to make the model understand the sentence context even further since the natural text grammar is not always straightforward for the machine to know which words shall it pay attention to when doing the translation. For Example,

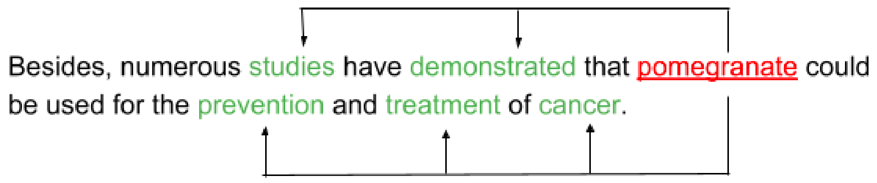

Besides, numerous studies have demonstrated that pomegranate could be used for the prevention and treatment of cancer.

Figure 3.5. *Example for demonstrating the importance of multi-head attention*

When the model sees the word "pomegranate" from this sentence during the translation process, there are many possible relations that the model should rather pay attention to with some weight differences. If the model has only one attention head for the prediction, it can be too subjective to conclude the correct weights and should have more attention layers to give more information.

Feed-Forward Layers

$$Feed - Forward(x) = max(0, xW_1 + b_1)W_2 + b_2 \tag{3.4}$$

The feed-forward network is applied in both encoder and decoder layers. It is position-wise, which means it is applied to each input vectors separately. The feed-forward consists of two linear transformations and a ReLu activation function.

Positional Encoding

$$PE_{(pos,2i)} = sin(\frac{pos}{1000^{2i/d_{model}}}) \tag{3.5}$$

$$PE_{(pos,2i+1)} = cos(\frac{pos}{1000^{2i/d_{model}}}) \tag{3.6}$$

The positional encoding is proposed to add the feature of word position within the sentence into the model. It has the same dimension as the model input (d_{model}), so it can be summed directly with the input embedding vector. This process is done at the beginning of the model initialization stage after embedding the input and the target sentences into word vectors. The value of the positional encoding vector is calculated from the sine and the cosine function following the equation 3.6. pos refers to the word position, d is the dimension of the model embedding input, and i is the iteration value of the positional encoding vector dimension.

In the Transformer original paper by Vaswani et al. 2017, the number of heads in Multi-head attention (h) is set to be 8. The dimensions of query, key, and value vectors are set to be 64, which is the number of model input embedding dimensions divided by the number of heads ($512/8 = 64$). This is to make the total computational cost similar to one head attention with 512 Q, K, V dimensions. The transformer model is trained with the standard WMT English-German dataset with 4.5 million sentence pairs using byte-pair encoding with 37,000 token vocabularies. It is trained with Adam optimizer with $\epsilon = 1e^{-9}$, $\beta_1 = 0.9$, and $\beta_2 = 0.98$. The learning rate varies warm up over 4,000 steps with L2 weight decay of 0.5.

An issue with the transformer model is it can only receive the fixed length, 512 dimensions, word embedding input. If the input has a longer length than that, it will only take the first 512 tokens and leave the rest tokens unused. This gave an intuition of another transformer-based model, TransformerXL(Dai et al. 2019) which can be trained by the longer sentences which can solve the context fragmentation problem.

3.1.3 BERT

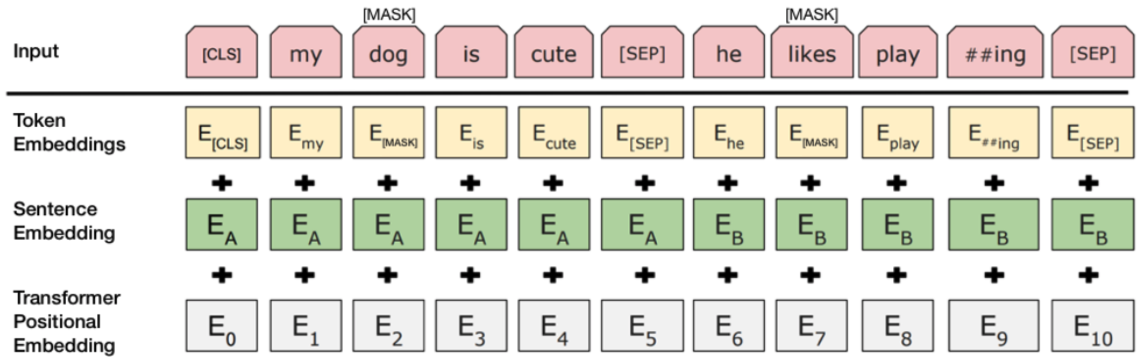
BERT (Devlin et al. 2019) is a machine learning model for language understanding introduced by Google in 2018. It is the model for understanding the natural language in a deep context. It has been pre-trained into many versions by researchers from many domains. The model structure was evolved from the deep learning model, Transformers. BERT is shorted for "Bidirectional Encoder Representation from Transformers". Its architecture is the transformer model's encoders stacked on top of each other. It effectively addresses ambiguity in a human-like common sense which is the most challenging task for Natural Language Processing. In 2019, Google announced using BERT in their production for their search engine.

In the original paper by Devlin et al. 2019, there were two trained BERT models introduced, BERT-based and BERT-Large. The BERT-based model has 12 encoder unit layers, 768 hidden units in feedforward networks, and 12 attention heads. BERT-Large has 24 encoder layers, 1024 hidden size, and 16 attention heads. The BERT-base model has the exact same parameter with GPT(Radford et al. 2018) by OpenAI for the performance comparison purpose. While BERT-base is bidirectional where the attention can be calculated across all words in the sentence, the GPT is constructed by decoder layers that omit access to the context on the right of the sentences.

Regarding the bidirectional feature of BERT compared to other language models that have been introduced before, BERT shows that its bidirectional training technique can have the model understand the context in a deeper context than having a single direction training or the concatenation of results from the left-to-right and right-to-left pieces of training, which is the concept proposed in Peters et al. 2018. BERT is non-directional where all word embedding vectors can be trained simultaneously, and the contextual relations can be referred to any other words in the input sentence from both left and right sides. The training techniques that BERT has introduced are the masked language model (MLM) and the next sentence prediction (NSP).

Input Representation

The input of BERT is constructed from three vectors, a token embedding, a segment embedding, and a position embedding following the Figure 3.6. According to Devlin et al. 2019, the token embedding is tokenized by the pretrained embedding, WordPiece embedding Wu et al. 2016 where it stores a 30,000 token vocabulary. The beginning of the input sequence is added with a token [CLS], and a token [SEP] is added between each sentence in the input sequence. The segment embedding is the label for separating the different sentences within one input vector. Position embedding is the position of the words in the input sequences. The requirement for having the segment embedding and the position embedding is to reserve the word order features during the training process. By summing the token embedding, segment embedding, and position embedding together, we get the input for BERT.



Input
[CLS]
my
[MASK]
dog
is
cute
[SEP]
he
[MASK]
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} $E_{[MASK]}$ E_{is} E_{cute} $E_{[SEP]}$ E_{he} $E_{[MASK]}$ E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Sentence Embedding
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Transformer Positional Embedding
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Figure 3.6. *Input Representation (Devlin et al. 2019).*

Pre-training

In this stage, BERT is trained to understand the language and the context by these two unsupervised tasks.

Masked Language Model (MLM)

When training BERT, 15% of words in each input embedding sequence are randomly replaced with [MASK] token. It has to predict the [MASK] words based on the 85% of left words in the sequence. This is similar to the cloze test in human language learning, where we are given a task to fill in the blanks for checking our language context understanding. During the training process, the 15% of words are not always be replaced with [MASK] token, as doing so will prevent the words from being seen in the fine-tuning stage. Instead, they are 80% of the time be replaced with [MASK] token, 10% of the time be replaced by random words, and 10% has been left unchanged. The process started from passing the input encoding with [MASK] tokens, passed through transformer encoder layers and a classification layer in the end. The classification layer transforms the output vectors into a vocabulary dimension where it uses the softmax function to calculate the probability of each word in the vocabulary. Only the output prediction of [MASK] token is taken into consideration during the backpropagation. The cross-entropy loss function ignores other un-masked words.

Next Sentence Prediction (NSP)

During the pretraining process, BERT passed two sentences for the model to predict whether the second sentence is the subsequent sentence to the first sentence. 50% of the input sequences are pair sentences from the original document, while other 50% inputs have the second sentence randomly chosen from the document.

During the implementation, the two training tasks, MLM and NSP, are processed together, as can be seen in Figure 3.7. The final hidden vector of the [CLS] token is token as C and is used as an output for the NSP task. The original paper Devlin et al. 2019 uses text data from the BooksCorpus (Zhu et al. 2015) with 800 Million words, English Wikipedia with 2,500 Million words in the sentences level, and also incorporates document-level input from Billion Word Benchmark (Chelba et al. 2013).

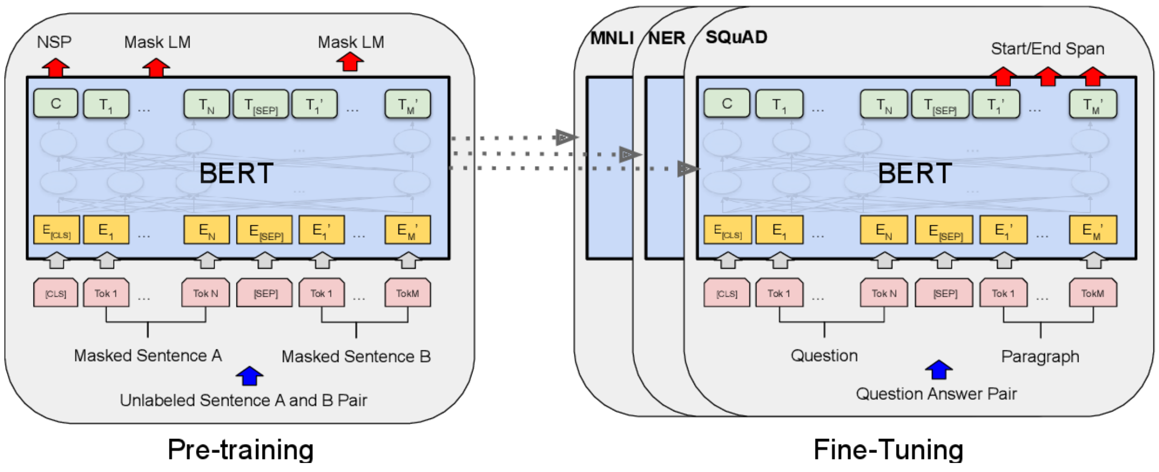


Figure 3.7. *BERT Training (Devlin et al. 2019).*

Fine-tuning

At this point, BERT can be fine-tuned to do different downstream tasks, such as question and answering, named-entity recognition, and many other classification tasks, for instance, sentiment analysis, relation extraction, etc. The input and output layers are used in different forms depending on the task we would like to achieve. The input can be two sentences packed together for sentence pairs, paraphrasing, question answering, or any other tasks that require the language understanding of relations between sentences. Additionally, the input embedding can also be just a one single sentence and be trained for classification and entity recognition tasks.

Specifically, in this research, we aim to use BERT for relation extraction between the two annotated tokens, @FOOD$ and @DISEASE$. Our fine-tuning stage is similar to the training of the Stanford Sentiment Treebank (SST-2) and the corpus of linguistic acceptability (CoLA). They are parts of the tasks in GLUE benchmark experiments (A. Wang et al. 2018) which the original paper(Devlin et al. 2019) has conducted. SST-2 is the task for classifying the sentiment of single-sentence input from movie reviews. CoLA is the task predicting the linguistic acceptability of single-sentence inputs. With these specific types of classification tasks, the classification layer is added to the BERT output for the predicting the [CLS] token. The illustration can be seen in the Figure 3.8

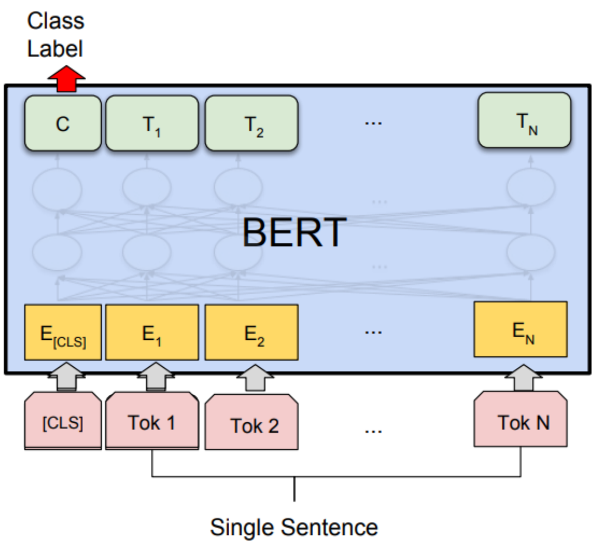


Figure 3.8. *Single Sentence Classification Fine-Tuning Task (Devlin et al. 2019).*

The training time in this fine-tuning stage is relatively fast since it only has to learn the new output parameters in the output layer, but the rest of the model parameters are only fine-tuned.

3.1.4 BioBERT

BioBERT (Lee et al. 2019) is one of the variations of BERT where the BERT model was pre-trained with Biomedical text data from PubMed abstracts and PMC articles. The pre-trained BioBERT model can be fine-tuned to do specific natural language processing tasks in the biomedical area.

The key idea of BioBERT is to give the BERT model specialization for understanding the specific domain knowledge, the biomedical field. Since the language used in a specialized field has a lot of terms and jargon that make it only understandable to the specialists but hard to fathom for ordinary people. BERT model does not bring the best out of biomedical text data since BERT has been trained for general language purposes. BioBERT, which has been pretrained with biomedical text papers, is expected to perform better with specialized natural language processing tasks in biomedical text data.

The work by (Lee et al. 2019) has pre-trained BERT not only with biomedical text but also tried the different combinations with general language text from English Wikipedia and BooksCorpus dataset. The datasets used in this work are detailed as follows.

Corpus	Number of Words
English Wikipedia	2.5 Billion
BooksCorpus	0.8 Billion
PubMed Abstracts	4.5 Billion
PMC Full-text Articles	13.5 Billion

The work provides four BERT fine-tuning models in comparison by the different text corpora. Firstly, Lee et al. 2019 initialized BioBERT with BERT pretained with English Wikipedia and Books Corpus datasets. Then, they trained that BERT pretained model into BioBERT with the Biomedical data in combination as follows.

- Wiki + Book + PubMed
- Wiki + Book + PMC
- Wiki + Book + PubMed + PMC

The input embedding vectors for training the model is still the WordPiece embedding Wu et al. 2016, the same method as what BERT original paper (Devlin et al. 2019) used. The training process was done with batch sizes 10, 16, 32, 64. The learning rate was $5e^{-5}$, $3e^{-5}$, $1e^{-5}$ ran on NVIDIA V100 GPUs (32GB). In total, it takes 23 days of pre-training.

Then, the initialization of BERT model and the other three BioBERT models have been fine-tuned to do different tasks.

- Name Entity Recognition (NER)

The name entity recognition is a prevalent task in biomedical natural language processing. It is the task required for the prediction of domain-specific nouns in the biomedical text corpus. BERT architecture is proper for performing this task as it gives the probability prediction vector for the [MASK] word. Applying it for biomedical text nouns is an appropriate application for the task. The researchers used the preprocessed datasets designed for performing NER from X. Wang et al. 2018. The text data included many types of entities, such as proteins, genes, chemicals, and disease. They compared the result from BERT and BioBERT models with entity-level F1 score, precision, and recall for evaluating the result. In summary, BioBERT (pre-trained with PubMed and PMC) achieved the best performance and outperformed the state-of-art model for 0.62 by average F1-score.

- Relation Extraction

  The researchers have utilized the sentence classification from what has been introduced in BERT original paper (Devlin et al. 2019) using [CLS] token representation as to the classification output. In this work, [CLS] has been used for classifying the two different target entities in the input sentences. The input sentences are annotated with @GENES$ and @DISEASE$ for the BioBERT model to classify the relationship. The researchers incorporated the preprocessed RE datasets, GAD(Bravo et al. 2015), EU-ADR(Van Mulligen et al. 2012) and CHAMPROT(Krallinger et al. 2015) which the preprocessing step following the work by Lim and Kang 2018. The dataset used in this fine-tuning task includes the relations between gene-disease and protein-chemical. The result was reported with the measurements, F1-score, precision, and recall. In conclusion, BioBERT with PubMed, PMC pre-training outperforms the state-of-art model by 2.80 higher F1-score.

- Question Answering

  The researchers used the same fine-tuning architecture with BERT fine-tuning with SQuAD, one of the GLUE benchmark experiments dataset (A. Wang et al. 2018). The researchers used BioASQ dataset (Tsatsaronis et al. 2015) by incorporating the full abstracts and the pair questions and answers. On the final note, BioBERT pretrained with PubMed and PMC achieved the new state-of-art performance by the higher 7.0 of MRR score.

3.1.5 DistilBERT

DistilBERT(Sanh et al. 2020) is a light transformers model. It is one of the variations of the BERT model, which requires less resource during the pretraining stage, is 40% smaller in model size, and is faster for 60%. The architecture of DistilBERT is the same as BERT except that it removes the token-type embedding and the pooler. The number of layers also decreased by half. The crucial concept of this model is the knowledge distillation from the BERT model. This knowledge distillation of DistilBERT was created based on the work by Hinton et al. 2015 where it introduces the concept of knowledge distillation developed from the work of Buciluă et al. 2006 in 2006. This knowledge distillation is also known as teacher-student training, where the teacher refers to the larger model, BERT in this case, and the student is the smaller model, DistilBERT.

The loss in the model training process is calculated based on the cross-entropy from the teacher model instead of directly from the target class. This makes the distillation model gets supervision from the larger model.

$$L = -\sum_i t_i log s_i \tag{3.7}$$

t is the logits from the teacher model and s is logits from the student model

When optimizing the model, Kullback-Leibler loss is used as the loss function since it shows the probability distribution difference of the prediction probability outputs between the teacher and the student models. The probability distribution demonstrated in Hinton et al. 2015 is the Softmax function which is suitable for multiclass classification. It converts the logits, z_i, into a probability of each class the data belong to, p_i. T parameter in the Softmax function is set to 1 for the standard classification. However, it can be adjusted. The higher value of T leads to a smoother probability distribution between classes.

$$KL(p||q) = \sum_i p_i log(p_i) - \sum_i p_i log(q_i) \tag{3.8}$$

The Kullback-Leibler loss where p illustrates the probability output from the teacher model and q illustrates the probability output from the student model

$$p_i = \frac{exp(Z_i/T)}{\sum_j exp(Z_j/T)} \tag{3.9}$$

The Softmax function

According to Sanh et al. 2020, DistilBERT has experimented with General Language Understanding Evaluation (GLUE) benchmarks in comparison with BERT-base and ELMo models. Taking average values from each task score, DistilBERT scores 77, which is only 2.5 points behind the BERT-base model, moreover, higher than ELMo performance for 8.3 points.

Model	**Score**	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	WNLI
ELMo	68.7	44.1	68.6	76.6	71.1	86.2	53.4	91.5	70.4	56.3
BERT-base	79.5	56.3	86.7	88.6	91.8	89.6	69.3	92.7	89.0	53.5
DistilBERT	77.0	51.3	82.2	87.5	89.2	88.5	59.9	91.3	86.9	56.3

Table 3.1. *The DistilBERT performance comparison on GLUE benchmart (Sanh et al. 2020).*

Parenthetically, the knowledge distillation concept has also been applied to another transformer-based model, GPT2, called DistilGPT2.

3.2 Traditional Machine Learning

Traditional machine learning can be an approach for relation extraction tasks that can be done after applying some of the possible data extraction methods, such as word embedding and TF-IDF vectors (Ramos et al. 2003).

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

Since text data is not understandable by computer programming, transforming the text information into a numerical value is required. The concept behind TF-IDF is to vectorize the text documents into numerical value by the term frequency compared by how frequent the term is represented to the overall corpus.

TF-IDF calculation process starts from creating a corpus of all unique words in the document dataset and the frequency of their appearance in the dataset. The list of unique words is used as a dimension of the output vectors. The row will be the sentences in the text dataset.

TF is a fraction of the frequency of word occurrence divided by the number of words that the sentence contains.

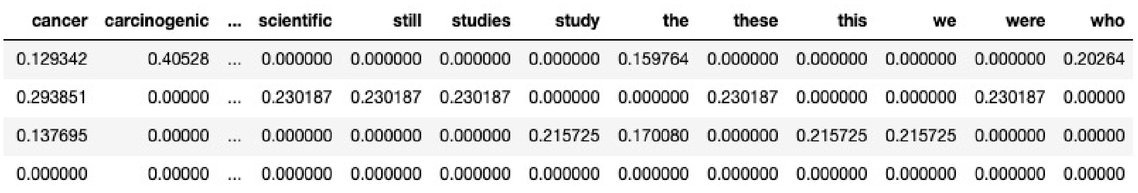
$$\text{TF} = \frac{\text{count of term x in sentence n}}{\text{total of words contain in sentence n}} \tag{3.10}$$

IDF is used for representing the importance of the terms. If the term occurs too much, it can be stopwords that do not convey important context to the document. IDF will weigh down the frequently used term and weigh up the scarce ones.

$$\text{IDF} = log(\frac{\text{N}}{\text{(occurence of term x in the document + 1)}}) \tag{3.11}$$

For Example, the following text data can extract 4 TF-IDF vectors with 48 dimensions as follow.

"The International Agency for Research on Cancer (WHO-IARC) classified red meat and processed meat as probably carcinogenic and carcinogenic for humans, respectively. These conclusions were mainly based on studies concerning colorectal cancer, but scientific evidence is still limited for other cancer locations. In this study, we investigated the prospective associations between red and processed meat intakes and overall breast and prostate cancer risk."



cancer	carcinogenic	...	scientific	still	studies	study	the	these	this	we	were	who
0.129342	0.40528	...	0.000000	0.000000	0.000000	0.000000	0.159764	0.000000	0.000000	0.000000	0.000000	0.20264
0.293851	0.00000	...	0.230187	0.230187	0.230187	0.000000	0.000000	0.230187	0.000000	0.000000	0.230187	0.00000
0.137695	0.00000	...	0.000000	0.000000	0.000000	0.215725	0.170080	0.000000	0.215725	0.215725	0.000000	0.00000
0.000000	0.00000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000

Figure 3.9. *The Demonstration of TF-IDF vectors generation*

After transforming the text dataset into TF-IDF vectors, traditional machine Learning can be used for the following relation extraction process.

3.2.2 Support Vector Machine (SVM)

Support Vector Machines (Noble 2006) is a nonprobabilistic supervised learning technique originally created for binary classification tasks. It can be used in regression either, but using this model as a classifier is more common. SVM beats other models most of the time when a high-dimensional dataset is proposed. This is because the model considers the set of attributes more than the number of data records.

To put it simply, this model separates two classes by drawing the best line separating the area with a boundary. The main question to be considered is what can be defined as the best line.

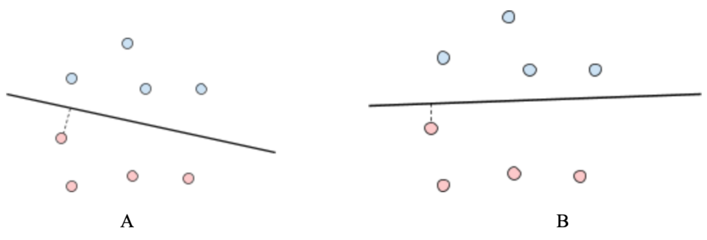


Figure 3.10. *An Illustration of SVM operation*

From the figure, when comparing between two lines which can separate the data into two classes, the line in picture A suits better since it is in the middle between 2 classes. On the other hand, line B is a bit too close to some points on both sides. We can see this clearly by seeing the distance between the closest point to the line. The distance between the closest point to the line from the figure is longer in picture A than in picture B. This is the principal concept demonstrating the SVM algorithm. It finds the separating line, which maximizes the distance of the line and the closest point.

Operation behind SVM

When the data has only two classes, the line will be the element that separates between two classes. With more variables, the process of separating data into different classes will become more complex. We call this the determination area where the line should be a draw as a decision boundary.

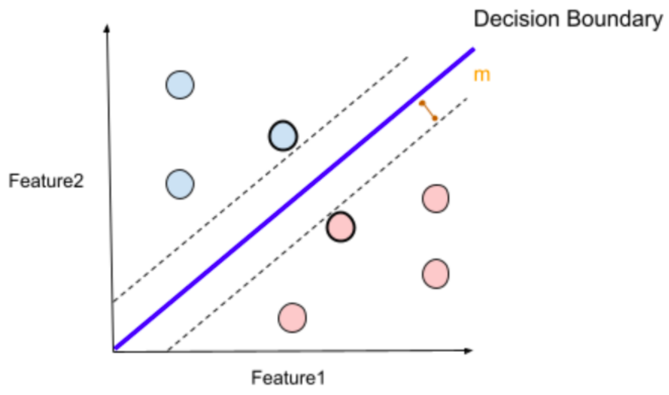


Figure 3.11. *SVM Decision Boundary*

The decision boundary is in the form of a hyperplane with a p-1 dimension when the dataset has p variables (Hastie et al. 2001). The figure 3.11 shows an example of a 1-dimensional hyperplane which is simply a line separating data points with two dimensions. The objects that fall on one side of the boundary will have the +1 as predicted value, while those that fall on another side will be -1.

$$y = sign(w^T x + b); y \in \{-1, 1\} \tag{3.12}$$

What can be defined as the best boundary for SVM is the line that maximizes the distance between the closest vectors (James et al. 2013). Vectors in this context refer to the data points which support the creation of the decision boundary. This concept co-aligns with its name, Support Vector Machines, referring to the vectors supporting the creation of the decision boundary (Rogers and Girolami 2016). The distance between the perpendicular decision boundary and the closest vectors on either side is the margin (m).

Let x_1 and x_2 be the closest points of two classes, the vector joining these two points will be $(x_1 - x_2)$. The direction of the points perpendicular to the decision boundary is $\frac{w}{||w||}$. The outcome of the inner product of these two terms will result in the margin from both sides combining.

$$2m = \frac{1}{||w||} w^T (x_1 - x_2) \tag{3.13}$$

We can derive m in the simpler form by reformatting the equation as follows (Rogers and

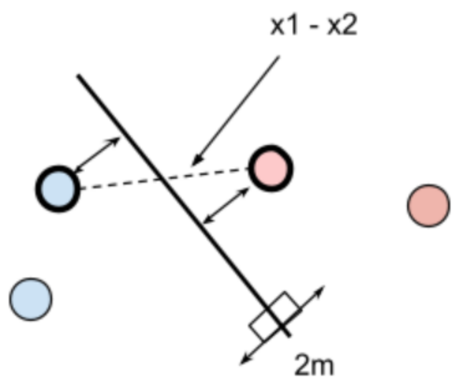


Figure 3.12. *A demonstration of the vector connecting x_1 and x_2*

Girolami 2016).

$$\begin{aligned}
2m &= w^T \frac{1}{||w||}(x_1 - x_2) \\
&= \frac{1}{||w||}(w^T x_1 - w^T x_2) \\
&= \frac{1}{||w||}(w^T x_1 + b - w^T x_2 - b) \\
&= \frac{1}{||w||}(1 + 1) \\
m &= \frac{1}{||w||}
\end{aligned} \tag{3.14}$$

Thus, to define the best hyperplane separating data into two classes, the goal is to maximize margin given that the line classifies the data into two classes when it falls in the area of margin edges and onwards. This optimization function can portray this concept.

$$\underset{w,b,||w||=1}{maximize} = \frac{1}{||w||} \tag{3.15}$$

Subject to $y_i(x_i^T w + b) \geq m$ for $i = 1, 2, 3, ..., N$

After that, one can optimize the equation with any convex optimization techniques, such as Stochastic Gradient descent and Lagrange multipliers methods.

Kernels Even though support vector machines can classify data in various cases, some cannot be separated linearly. In some cases, applying polynomial features as an additional dimension to the data can make it linearly separable (Géron 2017).

For instance, if our data have only one variable with two classes as follows.

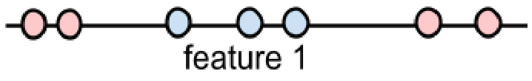


Figure 3.13. *A sample case suitable for applying Kernel function*

The algorithm cannot find the appropriate points for separating the two classes from one another. If we would like to apply a nonlinear function in linear regression, we can add more x variables with a coefficient. From the sample in figure 3.13, if we apply another feature with hyperbola function, $x_2 = x_1^2$, the classes can be separated by a linear decision boundary.

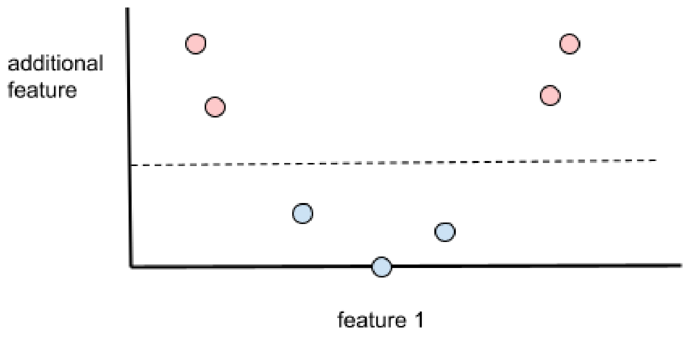


Figure 3.14. *Polynomial Kernel function*

There are many kernel functions available. The most well-known are these three following kernels.

Linear Kernel

$$k(x_n, x_m) = x_n^T x_m$$

Polynomial Kernel

$$k(x_n, x_m) = (1 + x_n^T x_m)^\gamma$$

Gaussian Kernel

$$k(x_n, x_m) = exp\{-\gamma(x_n - x_m)^T(x_n - x_m)\}$$

3.2.3 Naive Bayes

Naive Bayes is a classification machine learning algorithm that is well known for its fast performance (James et al. 2013). It is widely used in spam email detection, sentiment analysis, and recommendation system.

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)} \tag{3.16}$$

From the formula 3.16, the classifier tends to find the probability of c, given that x has occurred. The Naive Bayes assumes that all probability of events are independent from each other and equally affect the expected outcome. This is the reason for its name 'Naive' as in the actual world probability; there is more complex correlation and causation to take into account.

The term P(c|x) is the posterior probability, and the event P(c) is prior. Variable x represents the many possible events that can occur.

$$x = (x_1, x_2, x_3, ..., x_n)$$

Hence, we can expand the Naive Bayes formula by the chain's rule.

$$P(c|x_1, x_2, x_3, ..., x_n) = \frac{P(x_1|c)P(x_2|c)P(x_3|c)...P(x_n|c)P(c)}{P(x_1)P(x_2)P(x_3)...P(x_n)}$$

As for each possible event x, the denominator will remain the same. We can remove it to derive the approximate proportional calculation.

$$P(c|x_1, x_2, x_3, ..., x_n) \propto P(c)\Pi_{i=1}^{n}P(x_i|c)$$

In the case of multivariate classification, just like the relation extraction task, this research is achieving, the possible output c can be predicted by getting the class with the maximum value.

$$c = argmax_c P(c)\Pi_{i=1}^{n}P(x_i|c)$$

In the case of continuous variables input, the possible independent variable value is assumed to be followed the Gaussian distribution, and the formula for the conditional probability can be derived as follow.

$$P(c_i|x) = \frac{1}{\sqrt{2\pi\sigma_x^2}} exp(-\frac{(c_i - \mu_x)^2}{2\sigma_x^2})$$

3.3 Evaluation Method

After creating five different models, the test results are compared by two types of measurement, accuracy and F1 score.

To explain the measurement accuracy and F1 score, the concept of fundamental error is worth to be explored priorly. In every prediction task, many measurements indicate the effectiveness of the created models. The proper measurement should be appropriately chosen based on the specific case the model is trying to solve. According to the following table, there are four types of errors to consider in binary classification tasks.

		Prediction	
		True	False
Reality	True	True Positive (TP)	False Negative (FN)
	False	False Positive (FP)	True Negative (TN)

Table 3.2. *Four Types of Error*

The most used measurement for evaluating the machine learning model is accuracy which its calculation follows this formula.

Accuracy

$$\text{A} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.17}$$

Accuracy can be a powerful measurement that can tell how effective the model is. However, it can be unsuitable in many cases, depending on the task that the scientific method is trying to solve. For instance, in the case of the Covid-19 test, It will be more suitable to predict wrong for the negative case (False Positive) than to predict wrong for the positive case (False Negative) since the False Negative can lead to letting some sample patients struggle with the pandemic without knowing, being properly cured, and increase the possibility of spreading the virus to the others. The cost of False Negative, in this case, is hugely higher than False Positive and should maintain to be as low as possible. In this case, accuracy alone cannot be a suitable measurement, while using the recall measurement will be more appropriate(Emmert-Streib, Moutari et al. 2019).

Recall

$$\text{R} = \frac{TruePositive}{TruePositive + FalseNegative} \tag{3.18}$$

Recall, or sometimes has been called True Positive Rate or sensitivity, tells the rate of the correct, true prediction among all true samples in reality. The main factor affecting the recall is the False Negative value. With the high False Negative value, the recall will be less reflecting the low quality of the model as it cannot cover all positive cases(Emmert-Streib, Moutari et al. 2019).

Precision

$$\text{P} = \frac{TruePositive}{TruePositive + FalsePositive} \tag{3.19}$$

Precision, which can be called Positive Predictive Value, conveys the correct, true prediction rate over the amount of all samples predicted as true. The lower precision represents the possibility of getting false alarms by the chosen statistic test(Emmert-Streib, Moutari et al. 2019).

F1-Score

$$\text{F-Score} = (1 + \beta^2)\frac{(Precision * Recall)}{(Precision + Recall)} \tag{3.20}$$

F-Score is the weight between precision and recall. The weight can be adjusted by the parameter β. When β = 0, F-Score will follow the value of precision. While when $\beta \to \infty$, F-Score will correspond to the recall. $F_1 - Score$ is when the β has been set to 1, giving the harmonic mean of precision and recall. F-Score is a good option when one seeks the balance between precision and recall and in case of imbalanced data (Emmert-Streib, Moutari et al. 2019).

K-Folds Cross Validation

Cross-Validation is the method of segmenting the data into K amount of folds for the model training process. The model will be trained and tested for K amount of time and be evaluated in the end by an average of measurement results. In each iteration, there will be K-1 number of training sets and one set for testing. The test set will be changed in each round until all of the datasets have been used as a test set in K training iterations.

The support vector machine and Naive Bayes classification have been evaluated by 10-fold cross-validation in this research. While the BioBERT, BERT, and DistilBERT model has been measured by 3-fold cross-validation due to the time and resource taken.

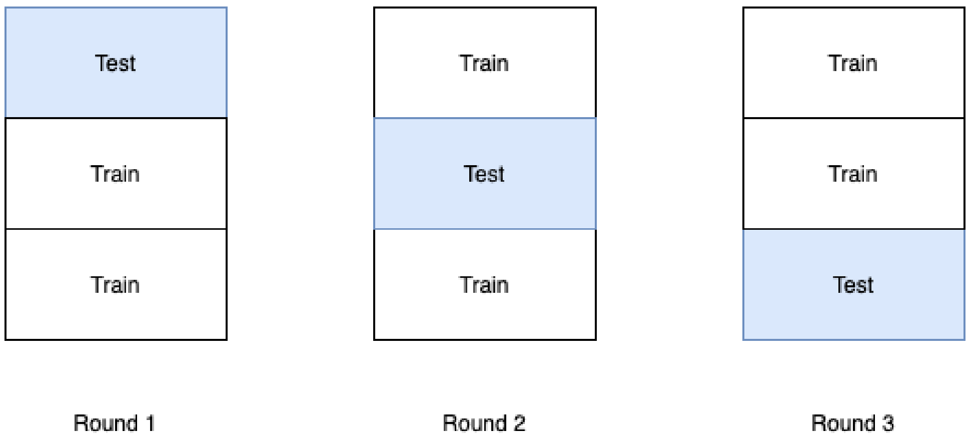


Figure 3.15. *An Illustration of 3-Fold Cross Validation*

Standard Error

In statistical analysis, we sample the dataset with the strategy to have the samples represent the overall population. After we got the samples, we created descriptive statistics to describe the sample data distribution. To measure how effective do the sample data represent the population distribution, we need to have the measurement for describing the deviation of the sample distribution from the population. The standard error (SE) is the measurement that estimates the deviation of the sample distribution by using the standard deviation (SE).

The main difference between SD and SE is that SD measures the deviation of the data points while SE measures how the mean deviates from the population mean.

$$\text{Standard Error} = \frac{\text{Standard Deviation}}{\sqrt{\text{Number of Samples}}} \tag{3.21}$$

$$\text{Standard Deviation} = \sqrt{\frac{\sum (x_1 - \bar{x})^2}{n-1}} \tag{3.22}$$

Learning Curve

Learning Curve (Emmert-Streib and Dehmer 2019) is a visualized method for diagnostic of the model performance. It shows the change in the prediction score when increasing the number of sample sizes. The way to get the most information from the curve is to conduct the learning curves comparing the learning of the training set and the validation set. These compared learning curves can tell two pieces of information about our created model. It tells the point where our model has a sufficient amount of samples for the training and how much bias and variance represent in the model. With an increase in sample size, the slope of the learning curve can be interpreted as follow.

- The learning significantly changes.
  This means the model still hasn't learned much from the given dataset enough to make an accurate prediction for the future dataset. It still requires a lot more data for the training process.
- The learning gradually changes.
  This shows that the model has almost reached the point that it can draw an accurate conclusion from the pattern in the given data. However, it still requires a lot more data to generalize the problem.
- The learning is flattened out.
  This means the sample size is sufficient.

In addition, the training and test learning curve in the graph can tell information on bias and variance that our created model has by seeing how the curve behaves.

- High Bias - Low Variance Suppose that the validation and the training learning curves converge to the prediction score that is quite low. It means that the model has a high bias. Moreover, if the training and validation curves have a small gap between each other, it means that the model generalizes well with the future unseen dataset that it can perform as well as the prediction performance of the training dataset. This indicates the low variance. The solution for high bias and low variance is to increase the complexity of the model so that it can fit more to the pattern in the dataset. We can describe this high bias, low variance that the model is underfitting.
- High Variance - Low Bias The high prediction score means the model has a low bias. Nonetheless, if the validation and training curves have a big gap between each other, that signifies the high variance value. In other words, the model is too overfitting to the training data set that it cannot generalize the task well enough when it encounters the other future datasets. This issue can be solved by increasing more sample size for the model to learn.

To illustrate and describe the learning curve, even more, we have created an example of the learning curves charts from two classification machine learning models, SVM and Random Forrest. We use an Iris dataset, a free open source dataset published by UCI Machine learning, to construct the learning curves.

As can be seen, 3.16 has a very small gap between the validation and training learning curves when compared to 3.17. This means the SVM model generalizes the data better while the random forest model has a higher variance. The random forest model has a higher tendency to be too overfitted with the training dataset. Both models perform well in terms of prediction scores; this shows that both models have low bias values.

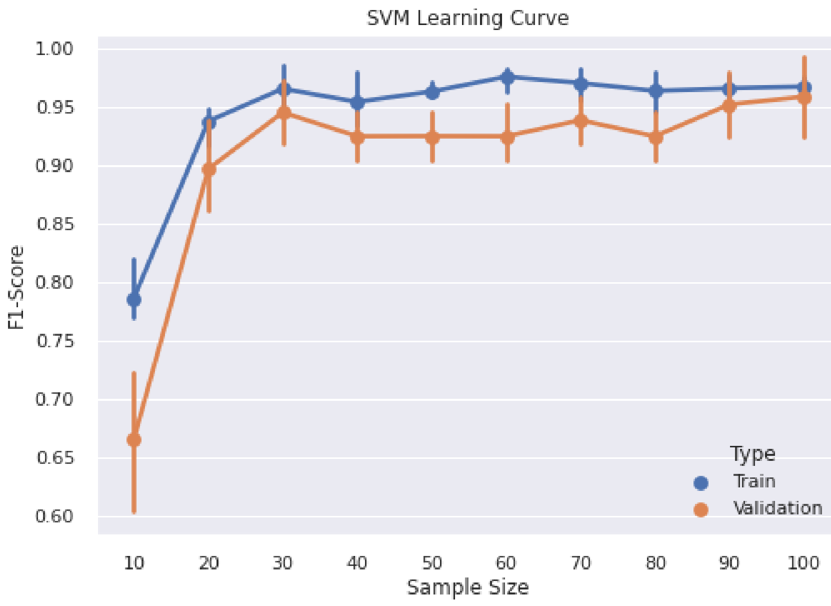


Figure 3.16. *SVM Learning Curve*

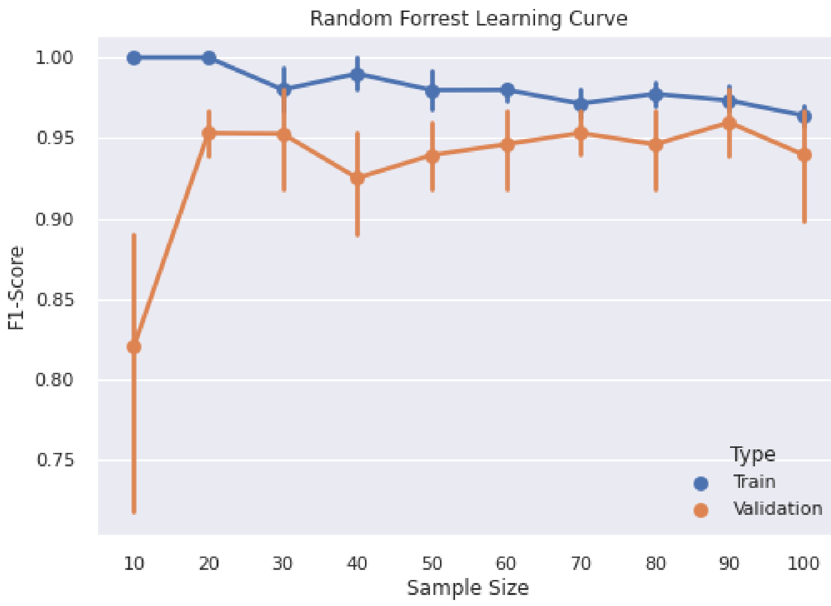


Figure 3.17. *Random Forest Learning Curve*

4 RESEARCH METHODS

4.1 Dataset

The dataset in this research has been acquired with a manual process from the biomedical text database. The process can be described as follow.

4.1.1 Data Collection

The dataset was acquired from PubMed, a free search engine for mainly accessing the MEDLINE database's research papers. It contains journals on the topic of life sciences and biomedical sciences. In this research, we focus on papers' content related to dietary compounds and cancer. Hence, search terms, 'food' and 'cancer' were incorporated accordingly. We have collected 150 papers in total.

The PubMed search engine can be accessed through the website, `https://pubmed.ncbi.nlm.nih.gov`

4.1.2 Data Preparation

The text data used in this research is the sentences from only the abstracts part of the papers. The sentences were split manually identified by a full stop(.). Then, the entity recognition process was done manually. If there existed words referring to the disease, cancer, they were replaced with '@DISEASE$'. Examples of these words can be 'cancer', 'bladder cancer', 'carcinoma'. Concurrently, if words referred to food compounds, they were replaced with '@FOOD$' annotation. Moreover, we created a field for collecting the relationship between entities within each sentence. If there appeared only one type of entity or no entity detected, we conclude that there wasn't any relationship appeared and the label was marked with the number 0.

If the @FOOD$ entity has an inhibit association to the @DISEASE$ entity, the label would be 2. For the provoked connection, we assigned a label equal to 3. Label 1 conveys the relation that two entities have an interconnection to each other but cannot tell the direction.

The number of records for each relation type can be reported as follows.

Label	Association
0	No Relation
1	Related
2	Prevent / Inhibit
3	Provoke

Table 4.1. *Relation Labels*

Label	Number of sentences	% Total
0	745	62.50%
1	107	8.98%
2	295	24.75%
3	45	3.77%

Altogether, we got 1,192 data records prepared for the model training.

4.2 Tools

In this research, the transformer-based model, BioBERT, BERT, and DistilBERT, had been implemented by the Transformers python library provided by the company, Hugging Face. The conventional machine learning models, SVM, and Naive Bayes was implemented by the Scikit-learn library.

Transformers Library

Transformers is a Python library for Natural Language Understanding (NLU) and Natural Language Generation (NLG). The library is compatible with Pytorch, Tensorflow machine learning libraries. It supports the state-of-art models and hosts the model hub where researchers can upload their fine-tuned model directly for others to use or develop forward.

Scikit-learn

Scikit-learn is a Python library for machine learning tasks with many supported conventional machine learning algorithms, such as Support Vector Machine, Random Forrest, K-mean Clustering, and Decision Tree. This library was originally created in 2007 as a Google summer code project by David Cournapeau. Afterward, there have been many contributors joined and helped continuously develop this tool until now.

Google Colab

Concerning the programming environment, the issue with fine-tuning a large complex model like BERT is that it requires a lot of computing power and library version dependencies. To solve this issue, Google Colaboratory (Colab) was used as the main environment for conducting this research. Colab is the product from google hosting, an online Jupyter notebook service. Any users can execute their python code from their browser directly without any setup in their local machine. The most prominent feature of this notebook making it suitable for deep learning research is the free access to the limited amount of GPU and TPU resources.

4.3 Analysis Method

After the data had been prepared, we used the data in fine-tuning process for BioBERT, BERT, and DistilBERT. The language model was downloaded into the workspace from the model hub hosted by Huggingface website, huggingface.co/models. For the BERT model, the version which we used for the fine-tuning process is bert-base-cased that has been updated on September 6, 2021. BioBERT model is provided by Data Mining and Information System Lab, Korea University. In this research, we used the BioBERT version 1.1, which was latest updated on May 19, 2021. In addition, for DistilBERT, we use the model distilbert-base-uncased that was updated on August 30, 2021. Mainly, for the model fine-tuning stage, the implementation was done with Transformer and Pytorch libraries.

The process of Transformer based model fine-tuning can be divided into two parts; the tokenization part and the model fine-tuning part.

Tokenization

With the transformers library, one can transfer the knowledge from the pretrained tokenizer to be used for word embedding of the input data. This can be done with the method, Autotokenizer. We set the maximum input length of the tokenizer as 319, which is equal to the number of words in the longest sentences among the dataset. We applied the padding and chose the output as PyTorch tensors. The pretrained tokenizers were downloaded from the three pretrained BERT-base models that we have conducted.

- dmis-lab/biobert-v1.1
- bert-base-cased
- distilbert-base-uncased

Model Fine-tuning

After the data had been transformed into a numeric vector with the pretrained tokenizer, we downloaded the pre-trained weights of each BERT-based model to fine-tune with the prepared dataset. All models are available on the Huggingface hub. This step required a data transformation into PyTorch tensors. Since we use the Pytorch library for this training, we used the data loader function to train the model with a batch size equal to ten. Each BERT-based model was trained with three-fold cross-validation. In each fold, the model was trained with three epochs. The Adam optimizer was used with learning rate equal to $2e^{-5}$ and ϵ equal to $1e^{-8}$. All the models have been set to use the same output layers configuration as the GLUE task (A. Wang et al. 2018), Stanford Sentiment Treebank (SST-2), the dataset for fine-tuning classification for single sentence inputs.

Conventional Machine Learning Model

We started with the feature extraction process from the text data with the @DISEASE$ and @FOOD$ entities annotation. While we used the pre-trained tokenizer for deep learning methods, TF-IDF vectors were extracted from the input text in the conventional machine learning method. In the end, we got the input vectors with 3,573 dimensions as an input for our machine learning model. We created an SVM model with Linear Kernel to predict the target class. Another model we have constructed is the Gaussian Naive Bayes classifier. Both of the models are created by the Scikit-learn library and trained with ten-fold cross-validation.

4.4 Evaluation

After obtaining the models, we applied evaluation methods to justify the models' performance. For BERT, BioBERT, DistilBERT, we applied the three-Fold Cross Validation using accuracy and F1 score as the primary measurement. The reason behind choosing the K parameter equal to three is due to the computational and time consumption in fine-tuning process since Transformer models require lots of computational power in model tuning. We are setting K as three as it is considered not wasting too many resources and not too little to draw a conclusion based on the researcher's justification. The standard error is also recorded for measuring the reliability of the model performance in each fold.

As for SVM and Naive Bays, we used the same measurement but increased the K value to ten-Fold Cross-Validation since the conventional models' architecture is more straightforward and does not require too much resource.

In addition, after all results were acquired, we would like to get more information on the amount of training set required for the models. We have created the learning curves to visualize the effectiveness of the model learning process over the increasing sample size. We tried different training samples from 200, 400, 600, 800, 1000, 1192. The F1-Score from training and validation were compared.

We also constructed the learning curves for SVM and Naive Bayes to see the suitability in the number of training samples and the bias-variance trade-off in the models.

5 RESULTS

After we had done the process described in the method session, we evaluated the result using the evaluation method described to see how well the created models have performed. The process consists of the text data collection from the paper abstracts, data preparation by manually splitting the text into sentences level, annotating the @FOOD$ and @DISEASE$ entities, and extracting entities' relations. After that, we have created three BERT-based models from the prepared dataset. The models were fine-tuned with three-fold cross-validation with F1-score and accuracy recorded. The traditional machine learning models, SVM and Naive Bayes Classifiers, have also been constructed. In addition, we have created the learning curves to see how the five models behave with the increase in sample size.

5.1 BERT

With the entire 1,192 rows dataset, the fine-tuned BERT got F1-Score 0.7844 from the three-fold cross-validation with a standard error equal to 0.0092. However, when we measured by accuracy, the model got 0.8133 with 0.0027 standard error.

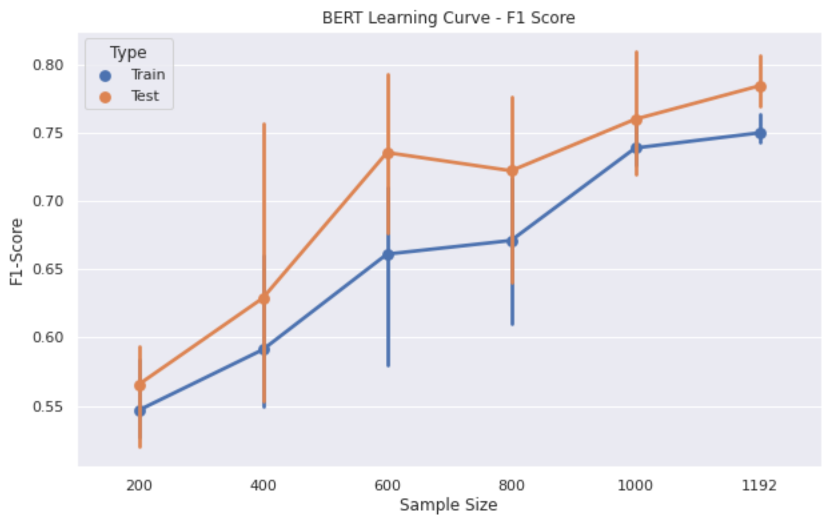


Figure 5.1. *BERT Learning Curve : F1-Score*

From plotting the learning curve, we can see that the model is still in the learning state as

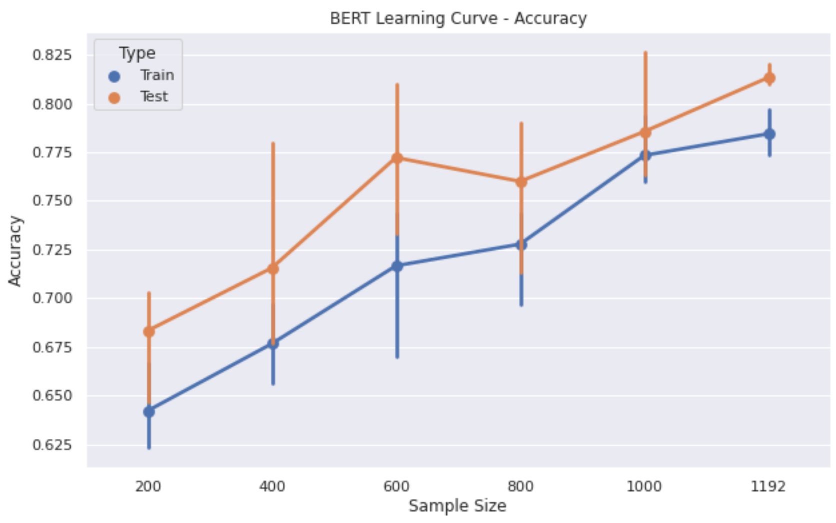


Figure 5.2. *BERT Learning Curve : Accuracy*

both test and training slope are still positive and tend to keep growing forward after our limited 1,192 records sample size. The model still has not reached to the optimum level that it can perform at its best.

5.2 DistilBERT

The fine-tuned DistilBERT got 0.7338 F1-Score with 0.0206 standard error. For accuracy, it received 0.7559 with 0.0160 standard error.

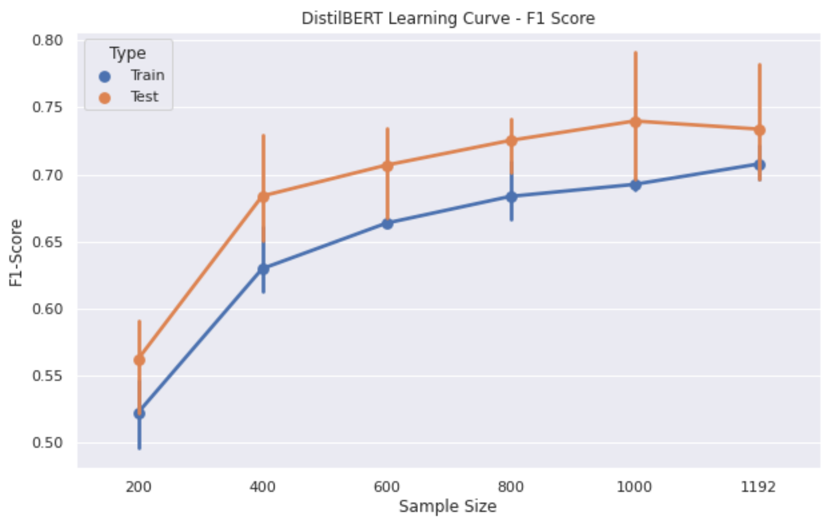


Figure 5.3. *DistilBERT Learning Curve : F1-Score*

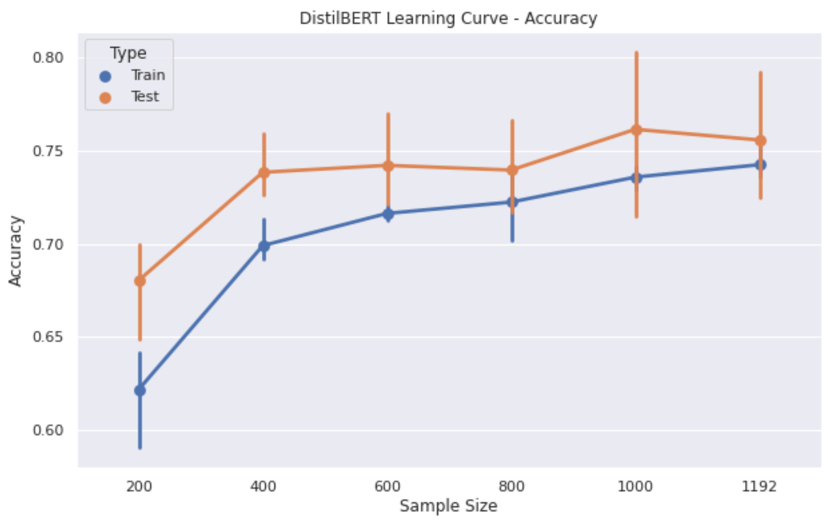


Figure 5.4. *DistilBERT Learning Curve : Accuracy*

When seeing the learning curve, we can see that the slope for DistilBERT is flatter than the BERT model showing that the sample size is almost enough for the model prediction. However, since the standard error for DistilBERT is higher than BERT at the 1,192 sample size, it shows that the flatten out learning curve might still fluctuate, and increasing the amount of sample size can make the model learn better.

Even though DistilBERT received less score than the fine-tuned BERT for 0.05, the fine-tuning process takes significantly less time. According to Sanh et al. 2019, DistilBERT is 40% smaller than the BERT model size, and it is 60% faster in fine-tuning time. From our observation table 5.1, we have recorded the time taken during the full dataset (1,192 records) trained in one epoch with a batch size equal to ten. The training set is 66.66%, and test set is 33.33% of the input dataset. While BERT took 1 minute and 26 seconds during fine-tuning one epoch with ten batch sizes, DistilBERT only took 43 seconds. That is 50% faster. This applied the same to the test stage; it took 16 seconds for BERT and 8 seconds for DistilBERT.

Model	Training	Test
BERT	0:01:26	0:00:16
DistilBERT	0:00:43	0:00:08
BioBERT	0:01:23	0:00:15

Table 5.1. *BERT-based Models Time Taken During Fine-tuning Stage*

5.3 BioBERT

The fine-tuned BioBERT received a 0.7855 F-score with 0.0123 standard error. The accuracy is 0.8164 with 0.0068 standard error.

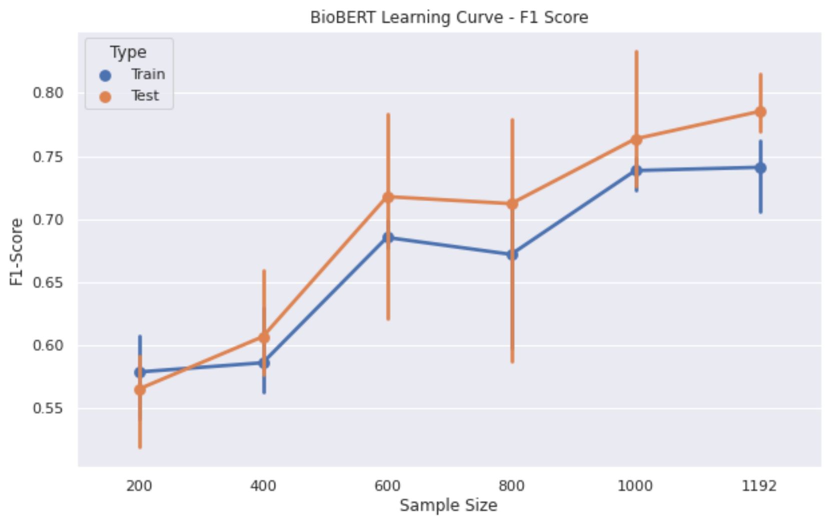


Figure 5.5. *BioBERT Learning Curve : F1-Score*

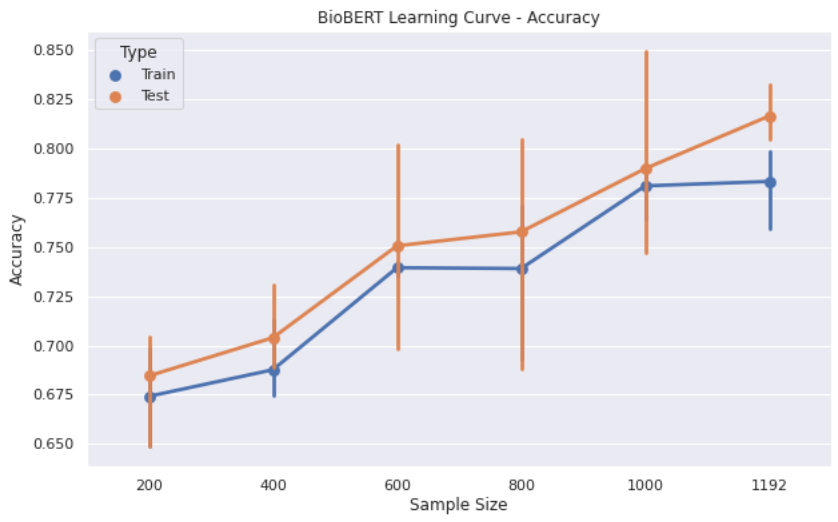


Figure 5.6. *BioBERT Learning Curve : Accuracy*

As can be seen, the learning curve has an upward trend with a positive slope showing that the learning can still go on even more with the larger amount of sample size. With the only amount of samples that we have, we can compare the measurement with the original BERT fine-tuned score that BioBERT performs better for 0.0011 F1-Score and 0.0031 for accuracy.

Since the difference between the BioBERT and BERT in our research is relatively small compared to the original paper(Lee et al. 2019) where BioBERT outperforms BERT in relation extraction task for 2.8 F1-score. Thus, we conducted the t-test statistics to compare the three-fold cross-validation test results from the two language models.

Model	Validatioin Set1	Validatioin Set2	Validation Set3	Mean	SD
BERT	0.8067	0.77	0.7766	0.7844	0.0195
BioBERT	0.8157	0.7695	0.7712	0.7855	0.0262

$$H_0 : \mu_{BioBERT} - \mu_{BERT} \leq d \tag{5.1}$$

$$H_1 : \mu_{BioBERT} - \mu_{BERT} > d \tag{5.2}$$

$$t = \frac{(\bar{x}_1 - \bar{x}_2) - (\mu_1 - \mu_2)}{\sqrt{\frac{S_1^s}{n_1} + \frac{S_2^s}{n_2}}} \tag{5.3}$$

$$df = \frac{(\frac{S_1^s}{n_1} + \frac{S_2^s}{n_2})^2}{\frac{(\frac{S_1^2}{n_1})^2}{n_1-1} + \frac{(\frac{S_2^2}{n_2})^2}{n_2-1}} \tag{5.4}$$

The probability value from this T-test equals to 0.4794, which is larger than 0.05. We can not reject that the average F1-Scores from BioBERT can be less than or equal to F1-Scores from BERT.

5.4 Support Vector Machine

We have conducted the traditional machine learning method, Support Vector Machine, with the TF-IDF vectors extracted from our text dataset. It got a 0.6497 F-Score with 0.0107 standard error. In terms of accuracy, it received 0.7139 with 0.0089 standard error.

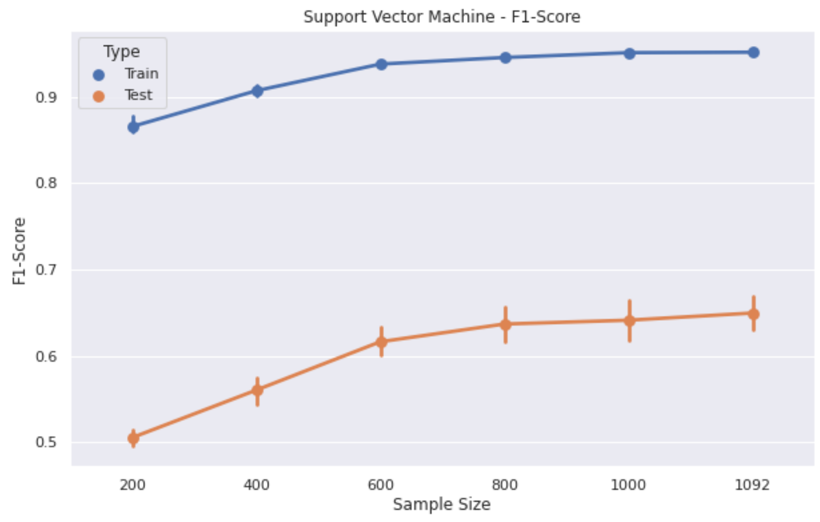


Figure 5.7. *SVM Learning Curve : F1-Score*

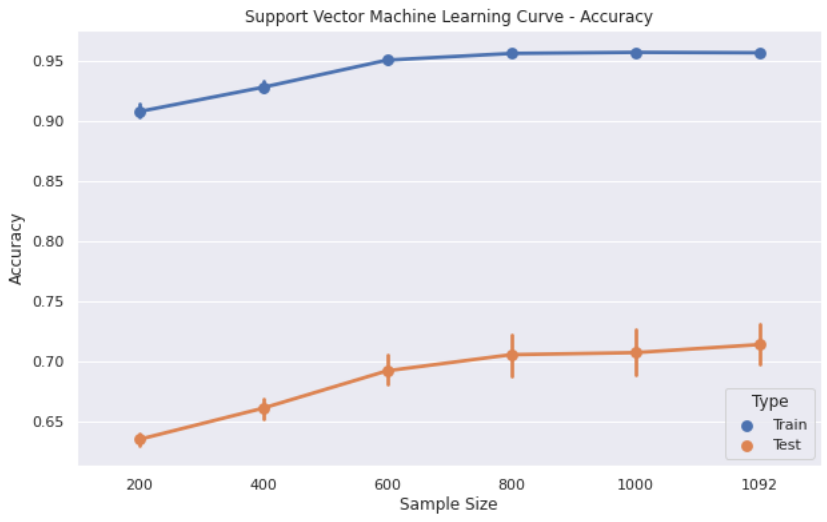


Figure 5.8. *SVM Learning Curve : Accuracy*

Looking at the SVM learning curve, we can see that the gap between the train and the test sets is very high. This shows that the model is overfitting to the training dataset that it cannot generalize the future dataset which it has not seen before that well. This type of model has high variance. Increasing the dataset can help the model to learn better.

5.5 Gaussian Naive Bayes

Gaussian Naive Bayes classifier received the worst result among other models. After we had trained the TF-IDF vectors with ten-fold cross-validation, we received a 0.5644 F-score with 0.0127 standard error and 0.5931 accuracy with 0.0132 standard error.

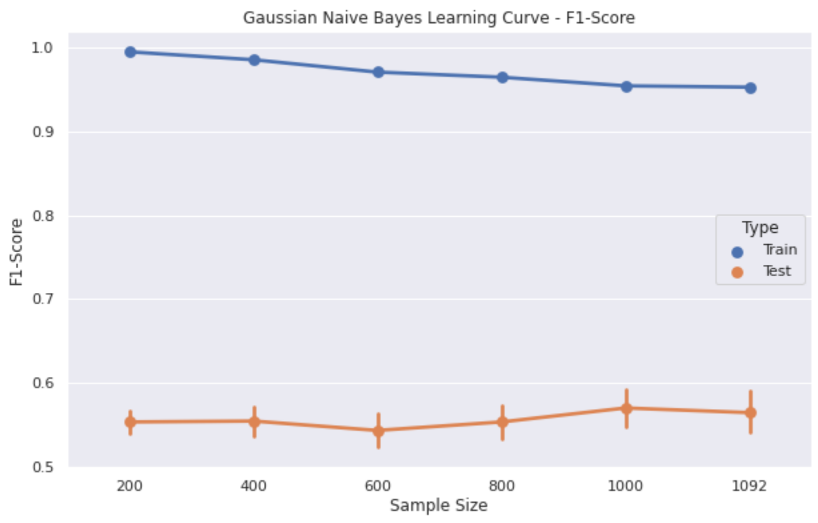


Figure 5.9. *Naive Bayes Learning Curve : F1-Score*

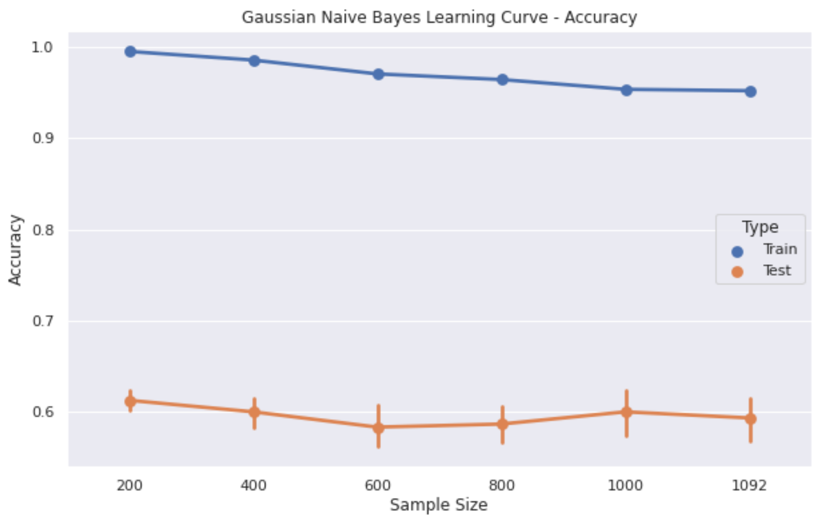


Figure 5.10. *Naive Bayes Learning Curve : Accuracy*

From plotting the the learning curves across the different sample size, we found that the learning didn't increase followed the increasing amount of samples overtime. The model has almost the same performance since the point that the sample size equal to 200. It can be said that the model has high bias error from its low prediction result which hasn't been improved over the growing amount of samples. In this case, increasing the model complexity is needed as the model isn't sophisticated enough to learn the pattern from the dataset.

5.6 The analysis of all results in comparison

Model	F1 Training	F1 Test	Accuracy Training	Accuracy Test
BERT	0.75(0.0054)	0.7844(0.0092)	0.7844 (0.0055)	0.8133 (0.0027)
DistilBERT	0.6296 (0.0127)	0.7338 (0.0206)	0.7426 (0.0046)	0.7559 (0.0160)
BioBERT	0.7412 (0.0144)	0.7855 (0.0123)	0.7832 (0.0098)	0.8164 (0.0068)
SVM	0.9517(0.0009)	0.6497(0.0107)	0.9564 (0.0007)	0.7139 (0.0089)
NB	0.9533(0.0009)	0.5644(0.0127)	0.9523(0.0009)	0.5931(0.0132)

The performance can be ranked in the following order.

BioBERT > BERT > DistilBERT > SVM > NB

We can conclude that the BERT-based language models have an exceptional performance compared to the traditional machine learning models. While BERT, the state-of-art language model, received a prominent performance. BioBERT has outperformed BERT by 0.0011 F1-Score.

To compare the learning process from each model even better, we have constructed the comparison in learning curves of the five constructed models.

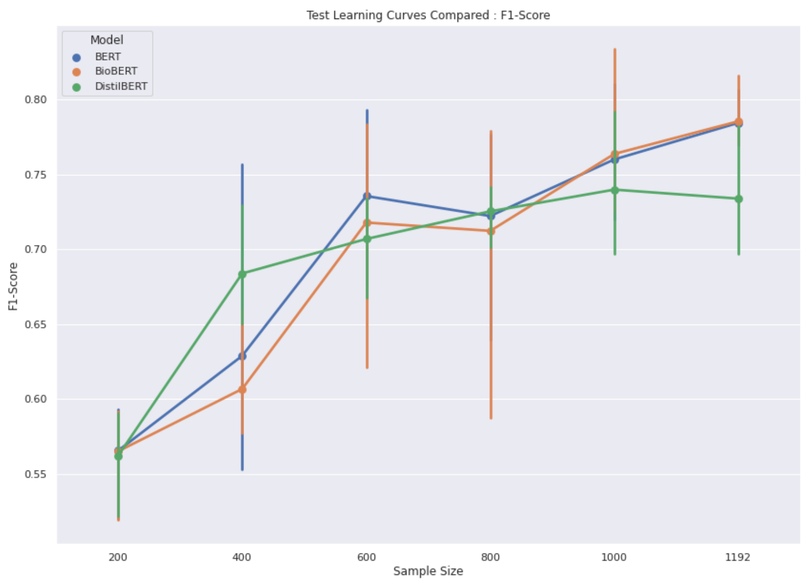


Figure 5.11. *BERT-based Models Learning Curve Comparison*

From the figure 5.11, we can see the BERT and BioBERT have steep learning curves and can continue the training process even more, to perform the task better. The DistilBERT learning curve, however, moderately increased after 1,000 samples. However, between 200 to 400 sample sizes, the DistilBert learning is steeper than the other two BERT-based models. This is due to the model architecture design of DistilBERT to be the lightweight model where it can learn faster but a bit poorer in performance than the full-scaled state-of-art model, BERT.

From the figure 5.12, we can see that the SVM has a better learning process compared to the Gaussian Naive Bayes, which almost has not learned from the newly increased sample size at all. At 200 sample size, Naive Bayes, however, perform better than the Support Vector Machine. Overall, this plot shows that SVM has a more sophisticated model that can learn from the increased data over time, while the Gaussian Naive Bayes' model architecture is too simple for performing the classification over TF-IDF vectors extracted from the text dataset.

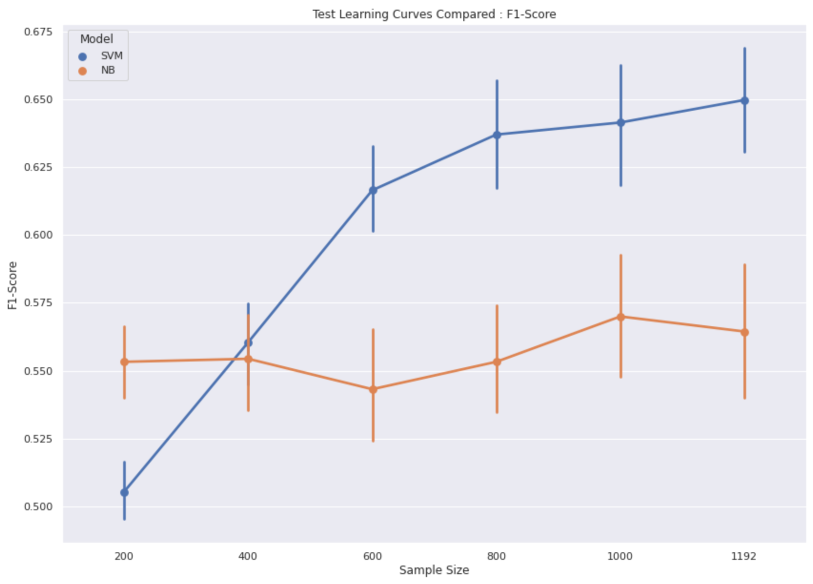


Figure 5.12. *SVM and NB leaning Curve Comparison*

6 CONCLUSION

To sum up, we have created the five machine learning models to classify the four types of relations between food and cancer entities from text data. The aim is to compare the suitability of the natural language processing models that are proper to be used as an information extraction system to construct a biomedical network knowledge graph.

The data was acquired from the biomedical papers search engine, PubMed. We only specify the search terms "food cancer" to get the relevant topic. We used the abstracts of the paper as our primary dataset. The sentences are separated by a full stop(.) and annotated with @FOOD$ and @DISEASE$ entities manually.

The prepared dataset was used for fine-tuning the three BERT-based language models, BERT, DistilBERT, BioBERT. The result proves that BioBERT is the best model choice for biomedical natural language text relation extraction from its domain specialization. It earned a 0.7855 F1-score with a 0.0123 standard error. The performance that it got is more than the original BERT fine-tuned model for 0.0011 F1-score. Even though the test results from our conducted three-fold cross-validation are not enough to statistically prove that the BioBERT got the better performance, increasing sample size in the future research will make the comparison of the results more apparent to see.

On the other hand, the DistilBERT has the least performance score among other BERT-based models. However, it is faster in fine-tuning. Particularly in our research with 1,192 input datasets, DistilBERT fine-tuning is 50% faster than BERT with only 0.05 lower in F1-score. It can be said that DistilBERT is more suitable for application production with a compact resource.

When seeing the learning curve of the three BERT-based models in comparison, it can be seen that BERT and BioBERT had a steep learning curve with the tendency to continue with more samples than 1,192, which we currently have. DistilBERT, however, tended to moderately increase with the higher sample size. At the smaller sample, DistilBERT learning curve was steeper than the other two models. This shows that with the limited amount of data, DistilBERT can be an option for an acceptable relation extraction performance. However, since the slope of DistilBERT still had the positive value, increasing more datasets will improve all BERT-based models' performance.

We have also conducted the traditional machine learning models, Support Vector Machine and Gaussian Naïve Bayes classifiers, as the baseline comparison to the proven state-of-art BERT-based models. The Support Vector Machine on the TF-IDF vectors performs tolerable prediction output with a 0.6497 F1-score. Naive Bayes, however, performs worst among other models. Seeing the learning curve in comparison between these two conventional models lets us understand the learning process that actually took place. The Naïve Bayes model had not learned from the increased dataset at all. It can be concluded that the model is not sophisticated enough to learn from the dataset as the F1-score for the test set was approximately around 0.56 since the beginning when the sample size equal to 200 and hadn't increased that much afterwards. Nonetheless, SVM also has an issue with the model overfitting with the training dataset. This can be seen from the large gap between the training and test curves. However, according to its test learning curve slope, which has an upward trend, this illustrates that it can be improved with significantly more samples in future work.

Despite the information gained from this research, this relation extraction task is only one of the tasks that will enable biomedical network creation. The other related tasks, for example, name-entity recognition, will need to be done to create an automated workflow. With the limitation in time and resource, we did the entity annotation by hand instead. Further research, study, and development will need to be done to construct this network that fully connects all the life science together.

This research shows the possible application of applying natural language processing in the specific domain, biomedical research, to create value for the field's specialists.

REFERENCES

Emmert-Streib, F., Dehmer, M. and Haibe-Kains, B. (2014). Gene regulatory networks and their applications: understanding biological and medical problems in terms of networks. *Frontiers in Cell and Developmental Biology* 2, 38. ISSN: 2296-634X. DOI: 10.3389/fcell.2014.00038. URL: https://www.frontiersin.org/article/10.3389/fcell.2014.00038.

Perera, N., Dehmer, M. and Emmert-Streib, F. (2020). Named Entity Recognition and Relation Detection for Biomedical Information Extraction. *Frontiers in Cell and Developmental Biology* 8, 673. ISSN: 2296-634X. DOI: 10.3389/fcell.2020.00673. URL: https://www.frontiersin.org/article/10.3389/fcell.2020.00673.

Abdul Wahab Muzaffar Farooque Azam, U. Q. (2015). A Relation Extraction Framework for Biomedical Text Using Hybrid Feature Set. *Computational and Mathematical Methods in Medicine* 2015, 12. DOI: 10.1155/2015/910423. URL: https://doi.org/10.1155/2015/910423.

Giuliano, C., Lavelli, A. and Romano, L. (Apr. 2006). Exploiting Shallow Linguistic Information for Relation Extraction from Biomedical Literature. *11th Conference of the European Chapter of the Association for Computational Linguistics*. Trento, Italy: Association for Computational Linguistics. URL: https://www.aclweb.org/anthology/E06-1051.

Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. (June 2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://www.aclweb.org/anthology/N19-1423.

Sanh, V., Debut, L., Chaumond, J. and Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR* abs/1910.01108. arXiv: 1910.01108. URL: http://arxiv.org/abs/1910.01108.

Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C. H. and Kang, J. (Sept. 2019). BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics* 36.4, 1234–1240.

Percha, B. ., Garten, Y. . and Altman, R. B. (2011). DISCOVERY AND EXPLANATION OF DRUG-DRUG INTERACTIONS VIA TEXT MINING. *Biocomputing 2012*, 410–421. DOI: 10.1142/9789814366496_0040.

Fundel, K. ., Kuffner, R. . and Zimmer, R. . (2006). RelEx–Relation extraction using dependency parse trees. *Bioinformatics* 23.3, 365–371. DOI: 10.1093/bioinformatics/btl616.

Yang, H. ., Swaminathan, R. ., Sharma, A. ., Ketkar, V. . and D'Silva, J. . (2011). Mining Biomedical Text towards Building a Quantitative Food-Disease-Gene Network. *Learning Structure and Schemas from Documents*, 205–225. DOI: 10.1007/978-3-642-22913-8_10.

Jensen, K. ., Panagiotou, G. . and Kouskoumvekaki, I. . (2014). Integrated Text Mining and Chemoinformatics Analysis Associates Diet to Health Benefit at Molecular Level. *PLoS Computational Biology* 10.1. Ed. by R. B. Altman, e1003432. DOI: 10.1371/journal.pcbi.1003432.

Emmert-Streib, F., Yang, Z., Feng, H., Tripathi, S. and Dehmer, M. (2020). An introductory review of deep learning for prediction models with big data. *Frontiers in Artificial Intelligence* 3, 4.

Liu, S. ., Tang, B. ., Chen, Q. . and Wang, X. . (2016). Drug-Drug Interaction Extraction via Convolutional Neural Networks. *Computational and Mathematical Methods in Medicine* 2016, 1–8. DOI: 10.1155/2016/6918381.

Sahu, S. K. and Anand, A. . (2018). Drug-drug interaction extraction from biomedical texts using long short-term memory network. *Journal of Biomedical Informatics* 86, 15–24. DOI: 10.1016/j.jbi.2018.08.005.

Shi, P. and Lin, J. (2019). *Simple BERT Models for Relation Extraction and Semantic Role Labeling*. arXiv: 1904.05255 [cs.CL].

Strubell, E., Ganesh, A. and McCallum, A. (2019). *Energy and Policy Considerations for Deep Learning in NLP*. arXiv: 1906.02243 [cs.CL].

MIT Technology Review (June 6, 2019). *Training a single AI model can emit as much carbon as five cars in their lifetimes*. URL: https://www.technologyreview.com/2019/06/06/239031/training-a-single-ai-model-can-emit-as-much-carbon-as-five-cars-in-their-lifetimes/.

TensorFlow Blog (May 18, 2020). *How Hugging Face achieved a 2x performance boost for Question Answering with DistilBERT in Node.js*. URL: https://blog.tensorflow.org/2020/05/how-hugging-face-achieved-2x-performance-boost-question-answering.html.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. and Polosukhin, I. (2017). *Attention Is All You Need*. arXiv: 1706.03762 [cs.CL].

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V. and Salakhutdinov, R. (2019). *Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context*. arXiv: 1901.02860 [cs.LG].

Radford, A., Narasimhan, K., Salimans, T. and Sutskever, I. (2018). Improving language understanding by generative pre-training.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K. and Zettlemoyer, L. (2018). *Deep contextualized word representations*. arXiv: `1802.05365 [cs.CL]`.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, Ł., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M. and Dean, J. (2016). *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*. arXiv: `1609.08144 [cs.CL]`.

Zhu, Y., Kiros, R., Zemel, R. S., Salakhutdinov, R., Urtasun, R., Torralba, A. and Fidler, S. (2015). Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books. *CoRR* abs/1506.06724. arXiv: `1506.06724`. URL: `http://arxiv.org/abs/1506.06724`.

Chelba, C., Mikolov, T., Schuster, M., Ge, Q., Brants, T. and Koehn, P. (2013). One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling. *CoRR* abs/1312.3005. arXiv: `1312.3005`. URL: `http://arxiv.org/abs/1312.3005`.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O. and Bowman, S. R. (2018). GLUE: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Wang, X., Zhang, Y., Ren, X., Zhang, Y., Zitnik, M., Shang, J., Langlotz, C. and Han, J. (2018). *Cross-type Biomedical Named Entity Recognition with Deep Multi-Task Learning*. arXiv: `1801.09851 [cs.IR]`.

Bravo, À., Piñero, J., Queralt-Rosinach, N., Rautschka, M. and Furlong, L. I. (2015). Extraction of relations between genes and diseases from text and large-scale data analysis: implications for translational research. *BMC bioinformatics* 16.1, 1–17.

Van Mulligen, E. M., Fourrier-Reglat, A., Gurwitz, D., Molokhia, M., Nieto, A., Trifiro, G., Kors, J. A. and Furlong, L. I. (2012). The EU-ADR corpus: annotated drugs, diseases, targets, and their relationships. *Journal of biomedical informatics* 45.5, 879–884.

Krallinger, M., Rabal, O., Leitner, F., Vazquez, M., Salgado, D., Lu, Z., Leaman, R., Lu, Y., Ji, D., Lowe, D. M. et al. (2015). The CHEMDNER corpus of chemicals and drugs and its annotation principles. *Journal of cheminformatics* 7.1, 1–17.

Lim, S. and Kang, J. (2018). Chemical–gene relation extraction using recursive neural network. *Database* 2018.

Tsatsaronis, G., Balikas, G., Malakasiotis, P., Partalas, I., Zschunke, M., Alvers, M. R., Weissenborn, D., Krithara, A., Petridis, S., Polychronopoulos, D. et al. (2015). An overview of the BIOASQ large-scale biomedical semantic indexing and question answering competition. *BMC bioinformatics* 16.1, 1–28.

Sanh, V., Debut, L., Chaumond, J. and Wolf, T. (2020). *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*. arXiv: `1910.01108 [cs.CL]`.

Hinton, G., Vinyals, O. and Dean, J. (2015). *Distilling the Knowledge in a Neural Network*. arXiv: `1503.02531 [stat.ML]`.

Buciluǎ, C. ., Caruana, R. . and Niculescu-Mizil, A. . (2006). Model compression. *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '06*. DOI: 10.1145/1150402.1150464.

Ramos, J. et al. (2003). Using tf-idf to determine word relevance in document queries. *Proceedings of the first instructional conference on machine learning*. Vol. 242. 1. Citeseer, 29–48.

Noble, W. S. (2006). What is a support vector machine?: *Nature biotechnology* 24.12, 1565–1567.

Hastie, T., Tibshirani, R. and Friedman, J. (2001). The elements of statistical learning. Springer series in statistics. *:* Springer.

James, G., Witten, D., Hastie, T. and Tibshirani, R. (2013). *An introduction to statistical learning*. Vol. 112. Springer.

Rogers, S. and Girolami, M. (2016). *A first course in machine learning*. Chapman and Hall/CRC.

Géron, A. (2017). Hands-on machine learning with scikit-learn and tensorflow: Concepts. *Tools, and Techniques to build intelligent systems*.

Emmert-Streib, F., Moutari, S. and Dehmer, M. (2019). A comprehensive survey of error measures for evaluating binary decision making in data science. *WIREs Data Mining and Knowledge Discovery* 9.5, e1303. DOI: https://doi.org/10.1002/widm.1303. eprint: https://wires.onlinelibrary.wiley.com/doi/pdf/10.1002/widm.1303. URL: https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.1303.

Emmert-Streib, F. and Dehmer, M. (2019). Evaluation of Regression Models: Model Assessment, Model Selection and Generalization Error. *Machine Learning and Knowledge Extraction* 1.1, 521–551. ISSN: 2504-4990. DOI: 10.3390/make1010032. URL: https://www.mdpi.com/2504-4990/1/1/32.

7 APPENDIX

All of the fine-tuned BERT-based models were uploaded to the huggingface model hub for any further development. They can be found at
`https://huggingface.co/Wiirin`

Moreover, the Python code used for constructing the learning curves from the fine-tuned BERT-based models that we have mentioned in this research is available at
`https://github.com/wiirin/BioBERT-finetuned-FoodCancer`

7.1 Measurement for Learning Curve

7.1.1 BERT

Sample Size	F1 Training	F1 Test	Accuracy Training	Accuracy Test
200	0.5467(0.0150)	0.5655(0.0187)	0.6422 (0.0105)	0.6833 (0.0150)
400	0.5911(0.0283)	0.6289(0.0524)	0.6767 (0.0094)	0.7156 (0.0265)
600	0.6611(0.0333)	0.7355(0.027)	0.7167 (0.0191)	0.7722 (0.0181)
800	0.6711(0.0259)	0.7222(0.0341)	0.7278 (0.0127)	0.7600 (0.0193)
1000	0.7389(0.0086)	0.76(0.0216)	0.7733 (0.0083)	0.7856 (0.0168)
1192	0.75(0.0054)	0.7844(0.0092)	0.7844 (0.0055)	0.8133 (0.0027)

7.1.2 DitilBERT

Sample Size	F1 Training	F1 Test	Accuracy Training	Accuracy Test
200	0.6296 (0.0127)	0.5622 (0.0170)	0.6217 (0.0129)	0.6803 (0.0129)
400	0.6636 (0.0002)	0.6839 (0.0192)	0.6991 (0.0058)	0.7385 (0.0086)
600	0.6836 (0.0109)	0.7069 (0.0164)	0.7164 (0.0020)	0.7422 (0.0118)
800	0.6925 (0.0018)	0.7253 (0.0099)	0.7225 (0.0089)	0.7397 (0.0118)
1000	0.7078 (0.0060)	0.7398 (0.0227)	0.7359 (0.0028)	0.7616 (0.0210)
1192	0.6296 (0.0127)	0.7338 (0.0206)	0.7426 (0.0046)	0.7559 (0.0160)

7.1.3 BioBERT

Sample Size	F1 Training	F1 Test	Accuracy Training	Accuracy Test
200	0.5787 (0.0158)	0.5651 (0.0188)	0.6742 (0.0117)	0.6846 (0.0146)
400	0.5861 (0.0179)	0.6068 (0.0217)	0.6877 (0.0104)	0.7040 (0.0110)
600	0.6855 (0.0055)	0.7179 (0.0402)	0.7394 (0.0018)	0.7506 (0.0244)
800	0.6721 (0.0302)	0.7123 (0.0510)	0.7390 (0.0192)	0.7577 (0.0288)
1000	0.7386 (0.0065)	0.7637 (0.0286)	0.7810 (0.0070)	0.7897 (0.0251)
1192	0.7412 (0.0144)	0.7855 (0.0123)	0.7832 (0.0098)	0.8164 (0.0068)

7.1.4 Support Vector Machine

Sample Size	F1 Training	F1 Test	Accuracy Training	Accuracy Test
200	0.8656(0.0056)	0.5054(0.0057)	0.9075 (0.0031)	0.6351 (0.0027)
400	0.9071(0.0034)	0.5604(0.0082)	0.9278 (0.0025)	0.6611 (0.0044)
600	0.9379(0.0018)	0.6165(0.0087)	0.9503 (0.0014)	0.6921 (0.0069)
800	0.9456(0.0012)	0.637(0.0106)	0.9559 (0.001)	0.7056 (0.009)
1000	0.9512(0.0019)	0.6414(0.0127)	0.9568 (0.0013)	0.7072 (0.0103)
1192	0.9517(0.0009)	0.6497(0.0107)	0.9564 (0.0007)	0.7139 (0.0089)

7.1.5 Gaussian Naive Bayes

Sample Size	F1 Training	F1 Test	Accuracy Training	Accuracy Test
200	0.9955(0.0005)	0.5533(0.0073)	0.9955(0.0005)	0.6124(0.0063)
400	0.9861(0.0011)	0.5544(0.0094)	0.986(0.0011)	0.5998(0.0088)
600	0.9713(0.0013)	0.5432(0.0114)	0.9708(0.0014)	0.583(0.0122)
800	0.9652(0.0011)	0.5533(0.0108)	0.9646(0.0011)	0.5864(0.0111)
1000	0.9549(0.0011)	0.57(0.0123)	0.9539(0.0012)	0.5998(0.013)
1192	0.9533(0.0009)	0.5644(0.0127)	0.9523(0.0009)	0.5931(0.0132)

7.2 Learning Curves Data

This table shows the detail on measurement in each fold cross validation from five conducted models. We have collected both accuracy and F1-Score.

n	Data	Type	Model	Measure
200	0.6367	Train	BERT	Accuracy
200	0.6667	Train	BERT	Accuracy
200	0.6233	Train	BERT	Accuracy
400	0.6767	Train	BERT	Accuracy
400	0.6967	Train	BERT	Accuracy
400	0.6567	Train	BERT	Accuracy
600	0.6700	Train	BERT	Accuracy
600	0.7367	Train	BERT	Accuracy
600	0.7433	Train	BERT	Accuracy
800	0.7433	Train	BERT	Accuracy
800	0.7433	Train	BERT	Accuracy
800	0.6967	Train	BERT	Accuracy
1000	0.7600	Train	BERT	Accuracy
1000	0.7667	Train	BERT	Accuracy
1000	0.7933	Train	BERT	Accuracy
1192	0.7733	Train	BERT	Accuracy
1192	0.7833	Train	BERT	Accuracy
1192	0.7967	Train	BERT	Accuracy

n	Data	Type	Model	Measure
200	0.7000	Test	BERT	Accuracy
200	0.6467	Test	BERT	Accuracy
200	0.7033	Test	BERT	Accuracy
400	0.6767	Test	BERT	Accuracy
400	0.7800	Test	BERT	Accuracy
400	0.6900	Test	BERT	Accuracy
600	0.7333	Test	BERT	Accuracy
600	0.8100	Test	BERT	Accuracy
600	0.7733	Test	BERT	Accuracy
800	0.7767	Test	BERT	Accuracy
800	0.7900	Test	BERT	Accuracy
800	0.7133	Test	BERT	Accuracy
1000	0.8267	Test	BERT	Accuracy
1000	0.7633	Test	BERT	Accuracy
1000	0.7667	Test	BERT	Accuracy
1192	0.8200	Test	BERT	Accuracy
1192	0.8100	Test	BERT	Accuracy
1192	0.8100	Test	BERT	Accuracy

n	Data	Type	Model	Measure
200	0.5300	Train	BERT	F1
200	0.5833	Train	BERT	F1
200	0.5267	Train	BERT	F1
400	0.5633	Train	BERT	F1
400	0.6600	Train	BERT	F1
400	0.5500	Train	BERT	F1
600	0.5800	Train	BERT	F1
600	0.6933	Train	BERT	F1
600	0.7100	Train	BERT	F1
800	0.7167	Train	BERT	F1
800	0.6867	Train	BERT	F1
800	0.6100	Train	BERT	F1
1000	0.7300	Train	BERT	F1
1000	0.7267	Train	BERT	F1
1000	0.7600	Train	BERT	F1
1192	0.7433	Train	BERT	F1
1192	0.7433	Train	BERT	F1
1192	0.7633	Train	BERT	F1
200	0.5833	Test	BERT	F1
200	0.5200	Test	BERT	F1
200	0.5933	Test	BERT	F1
400	0.5533	Test	BERT	F1
400	0.7567	Test	BERT	F1
400	0.5767	Test	BERT	F1
600	0.6767	Test	BERT	F1
600	0.7933	Test	BERT	F1
600	0.7367	Test	BERT	F1
800	0.7767	Test	BERT	F1
800	0.7500	Test	BERT	F1
800	0.6400	Test	BERT	F1
1000	0.8100	Test	BERT	F1
1000	0.7200	Test	BERT	F1
1000	0.7500	Test	BERT	F1
1192	0.8067	Test	BERT	F1
1192	0.7700	Test	BERT	F1
1192	0.7767	Test	BERT	F1

n	Data	Type	Model	Measure
200	0.6754	Train	BioBERT	Accuracy
200	0.6984	Train	BioBERT	Accuracy
200	0.6488	Train	BioBERT	Accuracy
400	0.7132	Train	BioBERT	Accuracy
400	0.6744	Train	BioBERT	Accuracy
400	0.6755	Train	BioBERT	Accuracy
600	0.7350	Train	BioBERT	Accuracy
600	0.7417	Train	BioBERT	Accuracy
600	0.7417	Train	BioBERT	Accuracy
800	0.7529	Train	BioBERT	Accuracy
800	0.7710	Train	BioBERT	Accuracy
800	0.6932	Train	BioBERT	Accuracy
1000	0.7638	Train	BioBERT	Accuracy
1000	0.7889	Train	BioBERT	Accuracy
1000	0.7901	Train	BioBERT	Accuracy
1192	0.7596	Train	BioBERT	Accuracy
1192	0.7912	Train	BioBERT	Accuracy
1192	0.7987	Train	BioBERT	Accuracy
200	0.7000	Test	BioBERT	Accuracy
200	0.6490	Test	BioBERT	Accuracy
200	0.7048	Test	BioBERT	Accuracy
400	0.7310	Test	BioBERT	Accuracy
400	0.6921	Test	BioBERT	Accuracy
400	0.6889	Test	BioBERT	Accuracy
600	0.7517	Test	BioBERT	Accuracy
600	0.8017	Test	BioBERT	Accuracy
600	0.6983	Test	BioBERT	Accuracy
800	0.7799	Test	BioBERT	Accuracy
800	0.8046	Test	BioBERT	Accuracy
800	0.6885	Test	BioBERT	Accuracy
1000	0.8495	Test	BioBERT	Accuracy
1000	0.7725	Test	BioBERT	Accuracy
1000	0.7471	Test	BioBERT	Accuracy
1192	0.8325	Test	BioBERT	Accuracy
1192	0.8124	Test	BioBERT	Accuracy
1192	0.8043	Test	BioBERT	Accuracy

n	Data	Type	Model	Measure
200	0.5873	Train	BioBERT	F1
200	0.6070	Train	BioBERT	F1
200	0.5418	Train	BioBERT	F1
400	0.6298	Train	BioBERT	F1
400	0.5625	Train	BioBERT	F1
400	0.5660	Train	BioBERT	F1
600	0.6804	Train	BioBERT	F1
600	0.6989	Train	BioBERT	F1
600	0.6772	Train	BioBERT	F1
800	0.7040	Train	BioBERT	F1
800	0.7140	Train	BioBERT	F1
800	0.5982	Train	BioBERT	F1
1000	0.7230	Train	BioBERT	F1
1000	0.7438	Train	BioBERT	F1
1000	0.7489	Train	BioBERT	F1
1192	0.7061	Train	BioBERT	F1
1192	0.7551	Train	BioBERT	F1
1192	0.7623	Train	BioBERT	F1
200	0.5844	Test	BioBERT	F1
200	0.5192	Test	BioBERT	F1
200	0.5918	Test	BioBERT	F1
400	0.6597	Test	BioBERT	F1
400	0.5836	Test	BioBERT	F1
400	0.5770	Test	BioBERT	F1
600	0.7487	Test	BioBERT	F1
600	0.7834	Test	BioBERT	F1
600	0.6215	Test	BioBERT	F1
800	0.7794	Test	BioBERT	F1
800	0.7700	Test	BioBERT	F1
800	0.5875	Test	BioBERT	F1
1000	0.8337	Test	BioBERT	F1
1000	0.7263	Test	BioBERT	F1
1000	0.7310	Test	BioBERT	F1
1192	0.8157	Test	BioBERT	F1
1192	0.7695	Test	BioBERT	F1
1192	0.7712	Test	BioBERT	F1

n	Data	Type	Model	Measure
200	0.5905	Train	DistilBERT	Accuracy
200	0.6413	Train	DistilBERT	Accuracy
200	0.6333	Train	DistilBERT	Accuracy
400	0.6918	Train	DistilBERT	Accuracy
400	0.7132	Train	DistilBERT	Accuracy
400	0.6922	Train	DistilBERT	Accuracy
600	0.7158	Train	DistilBERT	Accuracy
600	0.7125	Train	DistilBERT	Accuracy
600	0.7208	Train	DistilBERT	Accuracy
800	0.7021	Train	DistilBERT	Accuracy
800	0.7261	Train	DistilBERT	Accuracy
800	0.7392	Train	DistilBERT	Accuracy
1000	0.7294	Train	DistilBERT	Accuracy
1000	0.7412	Train	DistilBERT	Accuracy
1000	0.7371	Train	DistilBERT	Accuracy
1192	0.7365	Train	DistilBERT	Accuracy
1192	0.7537	Train	DistilBERT	Accuracy
1192	0.7375	Train	DistilBERT	Accuracy
200	0.7000	Test	DistilBERT	Accuracy
200	0.6490	Test	DistilBERT	Accuracy
200	0.6921	Test	DistilBERT	Accuracy
400	0.7298	Test	DistilBERT	Accuracy
400	0.7595	Test	DistilBERT	Accuracy
400	0.7262	Test	DistilBERT	Accuracy
600	0.7350	Test	DistilBERT	Accuracy
600	0.7700	Test	DistilBERT	Accuracy
600	0.7217	Test	DistilBERT	Accuracy
800	0.7349	Test	DistilBERT	Accuracy
800	0.7668	Test	DistilBERT	Accuracy
800	0.7173	Test	DistilBERT	Accuracy
1000	0.8034	Test	DistilBERT	Accuracy
1000	0.7667	Test	DistilBERT	Accuracy
1000	0.7147	Test	DistilBERT	Accuracy
1192	0.7921	Test	DistilBERT	Accuracy
1192	0.7505	Test	DistilBERT	Accuracy
1192	0.7250	Test	DistilBERT	Accuracy

n	Data	Type	Model	Measure
200	0.4959	Train	DistilBERT	F1
200	0.5459	Train	DistilBERT	F1
200	0.5250	Train	DistilBERT	F1
400	0.6125	Train	DistilBERT	F1
400	0.6607	Train	DistilBERT	F1
400	0.6157	Train	DistilBERT	F1
600	0.6638	Train	DistilBERT	F1
600	0.6638	Train	DistilBERT	F1
600	0.6632	Train	DistilBERT	F1
800	0.6661	Train	DistilBERT	F1
800	0.6748	Train	DistilBERT	F1
800	0.7099	Train	DistilBERT	F1
1000	0.6889	Train	DistilBERT	F1
1000	0.6923	Train	DistilBERT	F1
1000	0.6964	Train	DistilBERT	F1
1192	0.6969	Train	DistilBERT	F1
1192	0.7219	Train	DistilBERT	F1
1192	0.7045	Train	DistilBERT	F1
200	0.5909	Test	DistilBERT	F1
200	0.5216	Test	DistilBERT	F1
200	0.5741	Test	DistilBERT	F1
400	0.6716	Test	DistilBERT	F1
400	0.7293	Test	DistilBERT	F1
400	0.6506	Test	DistilBERT	F1
600	0.7190	Test	DistilBERT	F1
600	0.7339	Test	DistilBERT	F1
600	0.6677	Test	DistilBERT	F1
800	0.7329	Test	DistilBERT	F1
800	0.7415	Test	DistilBERT	F1
800	0.7016	Test	DistilBERT	F1
1000	0.7916	Test	DistilBERT	F1
1000	0.7312	Test	DistilBERT	F1
1000	0.6967	Test	DistilBERT	F1
1192	0.7820	Test	DistilBERT	F1
1192	0.7227	Test	DistilBERT	F1
1192	0.6968	Test	DistilBERT	F1

n	Data	Type	Model	Measure
200	1.0000	Train	NB	Accuracy
200	0.9950	Train	NB	Accuracy
200	0.9950	Train	NB	Accuracy
200	0.9950	Train	NB	Accuracy
200	0.9950	Train	NB	Accuracy
200	0.9950	Train	NB	Accuracy
200	0.9950	Train	NB	Accuracy
200	0.9950	Train	NB	Accuracy
200	0.9950	Train	NB	Accuracy
200	0.9950	Train	NB	Accuracy
400	0.9950	Train	NB	Accuracy
400	0.9875	Train	NB	Accuracy
400	0.9850	Train	NB	Accuracy
400	0.9825	Train	NB	Accuracy
400	0.9850	Train	NB	Accuracy
400	0.9850	Train	NB	Accuracy
400	0.9850	Train	NB	Accuracy
400	0.9850	Train	NB	Accuracy
400	0.9850	Train	NB	Accuracy
400	0.9850	Train	NB	Accuracy
600	0.9783	Train	NB	Accuracy
600	0.9750	Train	NB	Accuracy
600	0.9717	Train	NB	Accuracy
600	0.9650	Train	NB	Accuracy
600	0.9767	Train	NB	Accuracy
600	0.9683	Train	NB	Accuracy
600	0.9683	Train	NB	Accuracy
600	0.9683	Train	NB	Accuracy
600	0.9683	Train	NB	Accuracy
600	0.9683	Train	NB	Accuracy

n	Data	Type	Model	Measure
800	0.9675	Train	NB	Accuracy
800	0.9675	Train	NB	Accuracy
800	0.9638	Train	NB	Accuracy
800	0.9613	Train	NB	Accuracy
800	0.9725	Train	NB	Accuracy
800	0.9613	Train	NB	Accuracy
800	0.9613	Train	NB	Accuracy
800	0.9638	Train	NB	Accuracy
800	0.9638	Train	NB	Accuracy
800	0.9638	Train	NB	Accuracy
1000	0.9610	Train	NB	Accuracy
1000	0.9560	Train	NB	Accuracy
1000	0.9490	Train	NB	Accuracy
1000	0.9510	Train	NB	Accuracy
1000	0.9580	Train	NB	Accuracy
1000	0.9500	Train	NB	Accuracy
1000	0.9530	Train	NB	Accuracy
1000	0.9540	Train	NB	Accuracy
1000	0.9530	Train	NB	Accuracy
1000	0.9540	Train	NB	Accuracy
1192	0.9579	Train	NB	Accuracy
1192	0.9533	Train	NB	Accuracy
1192	0.9486	Train	NB	Accuracy
1192	0.9523	Train	NB	Accuracy
1192	0.9561	Train	NB	Accuracy
1192	0.9495	Train	NB	Accuracy
1192	0.9514	Train	NB	Accuracy
1192	0.9523	Train	NB	Accuracy
1192	0.9514	Train	NB	Accuracy
1192	0.9505	Train	NB	Accuracy

n	Data	Type	Model	Measure
200	0.6083	Test	NB	Accuracy
200	0.5833	Test	NB	Accuracy
200	0.5966	Test	NB	Accuracy
200	0.6471	Test	NB	Accuracy
200	0.6050	Test	NB	Accuracy
200	0.6218	Test	NB	Accuracy
200	0.5966	Test	NB	Accuracy
200	0.6050	Test	NB	Accuracy
200	0.6218	Test	NB	Accuracy
200	0.6387	Test	NB	Accuracy
400	0.6167	Test	NB	Accuracy
400	0.6083	Test	NB	Accuracy
400	0.6218	Test	NB	Accuracy
400	0.5966	Test	NB	Accuracy
400	0.5714	Test	NB	Accuracy
400	0.6303	Test	NB	Accuracy
400	0.6134	Test	NB	Accuracy
400	0.5462	Test	NB	Accuracy
400	0.5714	Test	NB	Accuracy
400	0.6218	Test	NB	Accuracy
600	0.6500	Test	NB	Accuracy
600	0.5417	Test	NB	Accuracy
600	0.6050	Test	NB	Accuracy
600	0.5798	Test	NB	Accuracy
600	0.5462	Test	NB	Accuracy
600	0.5966	Test	NB	Accuracy
600	0.6303	Test	NB	Accuracy
600	0.5378	Test	NB	Accuracy
600	0.5882	Test	NB	Accuracy
600	0.5546	Test	NB	Accuracy

n	Data	Type	Model	Measure
800	0.6333	Test	NB	Accuracy
800	0.5500	Test	NB	Accuracy
800	0.5630	Test	NB	Accuracy
800	0.5966	Test	NB	Accuracy
800	0.5294	Test	NB	Accuracy
800	0.6134	Test	NB	Accuracy
800	0.6387	Test	NB	Accuracy
800	0.5798	Test	NB	Accuracy
800	0.5798	Test	NB	Accuracy
800	0.5798	Test	NB	Accuracy
1000	0.6417	Test	NB	Accuracy
1000	0.5917	Test	NB	Accuracy
1000	0.5714	Test	NB	Accuracy
1000	0.6134	Test	NB	Accuracy
1000	0.5210	Test	NB	Accuracy
1000	0.6471	Test	NB	Accuracy
1000	0.6471	Test	NB	Accuracy
1000	0.6134	Test	NB	Accuracy
1000	0.5630	Test	NB	Accuracy
1000	0.5882	Test	NB	Accuracy
1192	0.6333	Test	NB	Accuracy
1192	0.5917	Test	NB	Accuracy
1192	0.5630	Test	NB	Accuracy
1192	0.5966	Test	NB	Accuracy
1192	0.5126	Test	NB	Accuracy
1192	0.6471	Test	NB	Accuracy
1192	0.6387	Test	NB	Accuracy
1192	0.6050	Test	NB	Accuracy
1192	0.5546	Test	NB	Accuracy
1192	0.5882	Test	NB	Accuracy

n	Data	Type	Model	Measure
200	1.0000	Train	NB	F1
200	0.9950	Train	NB	F1
200	0.9950	Train	NB	F1
200	0.9950	Train	NB	F1
200	0.9950	Train	NB	F1
200	0.9950	Train	NB	F1
200	0.9950	Train	NB	F1
200	0.9950	Train	NB	F1
200	0.9950	Train	NB	F1
200	0.9950	Train	NB	F1
400	0.9950	Train	NB	F1
400	0.9876	Train	NB	F1
400	0.9851	Train	NB	F1
400	0.9826	Train	NB	F1
400	0.9851	Train	NB	F1
400	0.9851	Train	NB	F1
400	0.9851	Train	NB	F1
400	0.9851	Train	NB	F1
400	0.9851	Train	NB	F1
400	0.9851	Train	NB	F1
600	0.9786	Train	NB	F1
600	0.9752	Train	NB	F1
600	0.9720	Train	NB	F1
600	0.9657	Train	NB	F1
600	0.9769	Train	NB	F1
600	0.9690	Train	NB	F1
600	0.9690	Train	NB	F1
600	0.9690	Train	NB	F1
600	0.9690	Train	NB	F1
600	0.9690	Train	NB	F1

n	Data	Type	Model	Measure
800	0.9680	Train	NB	F1
800	0.9679	Train	NB	F1
800	0.9644	Train	NB	F1
800	0.9619	Train	NB	F1
800	0.9728	Train	NB	F1
800	0.9619	Train	NB	F1
800	0.9619	Train	NB	F1
800	0.9645	Train	NB	F1
800	0.9645	Train	NB	F1
800	0.9645	Train	NB	F1
1000	0.9617	Train	NB	F1
1000	0.9569	Train	NB	F1
1000	0.9503	Train	NB	F1
1000	0.9521	Train	NB	F1
1000	0.9587	Train	NB	F1
1000	0.9511	Train	NB	F1
1000	0.9541	Train	NB	F1
1000	0.9551	Train	NB	F1
1000	0.9541	Train	NB	F1
1000	0.9549	Train	NB	F1
1192	0.9587	Train	NB	F1
1192	0.9542	Train	NB	F1
1192	0.9499	Train	NB	F1
1192	0.9533	Train	NB	F1
1192	0.9568	Train	NB	F1
1192	0.9506	Train	NB	F1
1192	0.9524	Train	NB	F1
1192	0.9533	Train	NB	F1
1192	0.9525	Train	NB	F1
1192	0.9516	Train	NB	F1

n	Data	Type	Model	Measure
200	0.5338	Test	NB	F1
200	0.5336	Test	NB	F1
200	0.5203	Test	NB	F1
200	0.5800	Test	NB	F1
200	0.5619	Test	NB	F1
200	0.5663	Test	NB	F1
200	0.5348	Test	NB	F1
200	0.5413	Test	NB	F1
200	0.5779	Test	NB	F1
200	0.5828	Test	NB	F1
400	0.5750	Test	NB	F1
400	0.5764	Test	NB	F1
400	0.5659	Test	NB	F1
400	0.5260	Test	NB	F1
400	0.5391	Test	NB	F1
400	0.5918	Test	NB	F1
400	0.5644	Test	NB	F1
400	0.5032	Test	NB	F1
400	0.5214	Test	NB	F1
400	0.5806	Test	NB	F1
600	0.5988	Test	NB	F1
600	0.5040	Test	NB	F1
600	0.5475	Test	NB	F1
600	0.5234	Test	NB	F1
600	0.5194	Test	NB	F1
600	0.5699	Test	NB	F1
600	0.5957	Test	NB	F1
600	0.5005	Test	NB	F1
600	0.5554	Test	NB	F1
600	0.5173	Test	NB	F1

n	Data	Type	Model	Measure
800	0.5933	Test	NB	F1
800	0.5093	Test	NB	F1
800	0.5294	Test	NB	F1
800	0.5639	Test	NB	F1
800	0.5119	Test	NB	F1
800	0.5948	Test	NB	F1
800	0.6022	Test	NB	F1
800	0.5388	Test	NB	F1
800	0.5427	Test	NB	F1
800	0.5464	Test	NB	F1
1000	0.6095	Test	NB	F1
1000	0.5495	Test	NB	F1
1000	0.5399	Test	NB	F1
1000	0.5688	Test	NB	F1
1000	0.5098	Test	NB	F1
1000	0.6265	Test	NB	F1
1000	0.6188	Test	NB	F1
1000	0.5835	Test	NB	F1
1000	0.5374	Test	NB	F1
1000	0.5563	Test	NB	F1
1192	0.6018	Test	NB	F1
1192	0.5447	Test	NB	F1
1192	0.5402	Test	NB	F1
1192	0.5542	Test	NB	F1
1192	0.5029	Test	NB	F1
1192	0.6239	Test	NB	F1
1192	0.6152	Test	NB	F1
1192	0.5781	Test	NB	F1
1192	0.5201	Test	NB	F1
1192	0.5633	Test	NB	F1

n	Data	Type	Model	Measure
200	0.9350	Train	SVM	Accuracy
200	0.9000	Train	SVM	Accuracy
200	0.9050	Train	SVM	Accuracy
200	0.9050	Train	SVM	Accuracy
200	0.9050	Train	SVM	Accuracy
200	0.9050	Train	SVM	Accuracy
200	0.9050	Train	SVM	Accuracy
200	0.9050	Train	SVM	Accuracy
200	0.9050	Train	SVM	Accuracy
200	0.9050	Train	SVM	Accuracy
400	0.9450	Train	SVM	Accuracy
400	0.9350	Train	SVM	Accuracy
400	0.9350	Train	SVM	Accuracy
400	0.9275	Train	SVM	Accuracy
400	0.9225	Train	SVM	Accuracy
400	0.9225	Train	SVM	Accuracy
400	0.9225	Train	SVM	Accuracy
400	0.9225	Train	SVM	Accuracy
400	0.9225	Train	SVM	Accuracy
400	0.9225	Train	SVM	Accuracy
600	0.9567	Train	SVM	Accuracy
600	0.9550	Train	SVM	Accuracy
600	0.9517	Train	SVM	Accuracy
600	0.9500	Train	SVM	Accuracy
600	0.9567	Train	SVM	Accuracy
600	0.9467	Train	SVM	Accuracy
600	0.9467	Train	SVM	Accuracy
600	0.9467	Train	SVM	Accuracy
600	0.9467	Train	SVM	Accuracy
600	0.9467	Train	SVM	Accuracy

n	Data	Type	Model	Measure
800	0.9563	Train	SVM	Accuracy
800	0.9563	Train	SVM	Accuracy
800	0.9513	Train	SVM	Accuracy
800	0.9550	Train	SVM	Accuracy
800	0.9600	Train	SVM	Accuracy
800	0.9513	Train	SVM	Accuracy
800	0.9538	Train	SVM	Accuracy
800	0.9575	Train	SVM	Accuracy
800	0.9588	Train	SVM	Accuracy
800	0.9588	Train	SVM	Accuracy
1000	0.9600	Train	SVM	Accuracy
1000	0.9560	Train	SVM	Accuracy
1000	0.9590	Train	SVM	Accuracy
1000	0.9550	Train	SVM	Accuracy
1000	0.9610	Train	SVM	Accuracy
1000	0.9560	Train	SVM	Accuracy
1000	0.9580	Train	SVM	Accuracy
1000	0.9610	Train	SVM	Accuracy
1000	0.9540	Train	SVM	Accuracy
1000	0.9480	Train	SVM	Accuracy
1192	0.9570	Train	SVM	Accuracy
1192	0.9589	Train	SVM	Accuracy
1192	0.9579	Train	SVM	Accuracy
1192	0.9542	Train	SVM	Accuracy
1192	0.9589	Train	SVM	Accuracy
1192	0.9533	Train	SVM	Accuracy
1192	0.9570	Train	SVM	Accuracy
1192	0.9589	Train	SVM	Accuracy
1192	0.9542	Train	SVM	Accuracy
1192	0.9542	Train	SVM	Accuracy

n	Data	Type	Model	Measure
200	0.6500	Test	SVM	Accuracy
200	0.6333	Test	SVM	Accuracy
200	0.6218	Test	SVM	Accuracy
200	0.6303	Test	SVM	Accuracy
200	0.6303	Test	SVM	Accuracy
200	0.6303	Test	SVM	Accuracy
200	0.6471	Test	SVM	Accuracy
200	0.6303	Test	SVM	Accuracy
200	0.6387	Test	SVM	Accuracy
200	0.6387	Test	SVM	Accuracy
400	0.6667	Test	SVM	Accuracy
400	0.6667	Test	SVM	Accuracy
400	0.6387	Test	SVM	Accuracy
400	0.6639	Test	SVM	Accuracy
400	0.6555	Test	SVM	Accuracy
400	0.6471	Test	SVM	Accuracy
400	0.6807	Test	SVM	Accuracy
400	0.6471	Test	SVM	Accuracy
400	0.6639	Test	SVM	Accuracy
400	0.6807	Test	SVM	Accuracy
600	0.6917	Test	SVM	Accuracy
600	0.6833	Test	SVM	Accuracy
600	0.6975	Test	SVM	Accuracy
600	0.7143	Test	SVM	Accuracy
600	0.6807	Test	SVM	Accuracy
600	0.6723	Test	SVM	Accuracy
600	0.7311	Test	SVM	Accuracy
600	0.6639	Test	SVM	Accuracy
600	0.6723	Test	SVM	Accuracy
600	0.7143	Test	SVM	Accuracy

n	Data	Type	Model	Measure
800	0.6833	Test	SVM	Accuracy
800	0.7083	Test	SVM	Accuracy
800	0.7143	Test	SVM	Accuracy
800	0.7227	Test	SVM	Accuracy
800	0.6975	Test	SVM	Accuracy
800	0.7227	Test	SVM	Accuracy
800	0.7479	Test	SVM	Accuracy
800	0.6555	Test	SVM	Accuracy
800	0.6723	Test	SVM	Accuracy
800	0.7311	Test	SVM	Accuracy
1000	0.6833	Test	SVM	Accuracy
1000	0.7083	Test	SVM	Accuracy
1000	0.7395	Test	SVM	Accuracy
1000	0.7227	Test	SVM	Accuracy
1000	0.7059	Test	SVM	Accuracy
1000	0.6891	Test	SVM	Accuracy
1000	0.7563	Test	SVM	Accuracy
1000	0.6555	Test	SVM	Accuracy
1000	0.6723	Test	SVM	Accuracy
1000	0.7395	Test	SVM	Accuracy
1192	0.7000	Test	SVM	Accuracy
1192	0.7083	Test	SVM	Accuracy
1192	0.7563	Test	SVM	Accuracy
1192	0.7227	Test	SVM	Accuracy
1192	0.7143	Test	SVM	Accuracy
1192	0.7143	Test	SVM	Accuracy
1192	0.7479	Test	SVM	Accuracy
1192	0.6639	Test	SVM	Accuracy
1192	0.6807	Test	SVM	Accuracy
1192	0.7311	Test	SVM	Accuracy

n	Data	Type	Model	Measure
200	0.9156	Train	SVM	F1
200	0.8537	Train	SVM	F1
200	0.8609	Train	SVM	F1
200	0.8609	Train	SVM	F1
200	0.8609	Train	SVM	F1
200	0.8609	Train	SVM	F1
200	0.8609	Train	SVM	F1
200	0.8609	Train	SVM	F1
200	0.8609	Train	SVM	F1
200	0.8609	Train	SVM	F1
400	0.9285	Train	SVM	F1
400	0.9194	Train	SVM	F1
400	0.9173	Train	SVM	F1
400	0.9063	Train	SVM	F1
400	0.8999	Train	SVM	F1
400	0.8999	Train	SVM	F1
400	0.8999	Train	SVM	F1
400	0.8999	Train	SVM	F1
400	0.8999	Train	SVM	F1
400	0.8999	Train	SVM	F1
600	0.9426	Train	SVM	F1
600	0.9446	Train	SVM	F1
600	0.9411	Train	SVM	F1
600	0.9352	Train	SVM	F1
600	0.9476	Train	SVM	F1
600	0.9379	Train	SVM	F1
600	0.9326	Train	SVM	F1
600	0.9326	Train	SVM	F1
600	0.9326	Train	SVM	F1
600	0.9326	Train	SVM	F1

n	Data	Type	Model	Measure
800	0.9450	Train	SVM	F1
800	0.9463	Train	SVM	F1
800	0.9421	Train	SVM	F1
800	0.9421	Train	SVM	F1
800	0.9502	Train	SVM	F1
800	0.9388	Train	SVM	F1
800	0.9436	Train	SVM	F1
800	0.9484	Train	SVM	F1
800	0.9497	Train	SVM	F1
800	0.9497	Train	SVM	F1
1000	0.9553	Train	SVM	F1
1000	0.9505	Train	SVM	F1
1000	0.9545	Train	SVM	F1
1000	0.9482	Train	SVM	F1
1000	0.9571	Train	SVM	F1
1000	0.9513	Train	SVM	F1
1000	0.9535	Train	SVM	F1
1000	0.9566	Train	SVM	F1
1000	0.9480	Train	SVM	F1
1000	0.9370	Train	SVM	F1
1192	0.9524	Train	SVM	F1
1192	0.9548	Train	SVM	F1
1192	0.9539	Train	SVM	F1
1192	0.9476	Train	SVM	F1
1192	0.9550	Train	SVM	F1
1192	0.9488	Train	SVM	F1
1192	0.9525	Train	SVM	F1
1192	0.9546	Train	SVM	F1
1192	0.9485	Train	SVM	F1
1192	0.9490	Train	SVM	F1

n	Data	Type	Model	Measure
200	0.5392	Test	SVM	F1
200	0.5180	Test	SVM	F1
200	0.4769	Test	SVM	F1
200	0.4976	Test	SVM	F1
200	0.4976	Test	SVM	F1
200	0.4873	Test	SVM	F1
200	0.5238	Test	SVM	F1
200	0.5015	Test	SVM	F1
200	0.5061	Test	SVM	F1
200	0.5061	Test	SVM	F1
400	0.5762	Test	SVM	F1
400	0.5832	Test	SVM	F1
400	0.5134	Test	SVM	F1
400	0.5781	Test	SVM	F1
400	0.5472	Test	SVM	F1
400	0.5289	Test	SVM	F1
400	0.5864	Test	SVM	F1
400	0.5452	Test	SVM	F1
400	0.5597	Test	SVM	F1
400	0.5864	Test	SVM	F1
600	0.6139	Test	SVM	F1
600	0.6076	Test	SVM	F1
600	0.6218	Test	SVM	F1
600	0.6556	Test	SVM	F1
600	0.6027	Test	SVM	F1
600	0.5838	Test	SVM	F1
600	0.6612	Test	SVM	F1
600	0.5879	Test	SVM	F1
600	0.5916	Test	SVM	F1
600	0.6384	Test	SVM	F1

n	Data	Type	Model	Measure
800	0.6070	Test	SVM	F1
800	0.6400	Test	SVM	F1
800	0.6451	Test	SVM	F1
800	0.6665	Test	SVM	F1
800	0.6304	Test	SVM	F1
800	0.6503	Test	SVM	F1
800	0.6905	Test	SVM	F1
800	0.5816	Test	SVM	F1
800	0.5975	Test	SVM	F1
800	0.6611	Test	SVM	F1
1000	0.6072	Test	SVM	F1
1000	0.6401	Test	SVM	F1
1000	0.6774	Test	SVM	F1
1000	0.6665	Test	SVM	F1
1000	0.6485	Test	SVM	F1
1000	0.6144	Test	SVM	F1
1000	0.7016	Test	SVM	F1
1000	0.5816	Test	SVM	F1
1000	0.5975	Test	SVM	F1
1000	0.6794	Test	SVM	F1
1192	0.6295	Test	SVM	F1
1192	0.6401	Test	SVM	F1
1192	0.6967	Test	SVM	F1
1192	0.6665	Test	SVM	F1
1192	0.6593	Test	SVM	F1
1192	0.6435	Test	SVM	F1
1192	0.6927	Test	SVM	F1
1192	0.5960	Test	SVM	F1
1192	0.6046	Test	SVM	F1
1192	0.6682	Test	SVM	F1