Lehrstuhl für Data Science

Comparing the Performance of NLP Toolkits and Evaluation measures in Legal Tech

Masterarbeit von

Muhammad Zohaib Khan

Supervised By: Prof. Dr. Jelena Mitrović

1. PRÜFER	2. PRÜFER
Prof. Dr. Jelena Mitrović	Prof. Dr. Michael Granitzer

March 23, 2021

Contents

Contents

Abstract

Recent developments in Natural Language Processing have led to the introduction of state-of-the-art Neural Language Models, enabled with unsupervised transferable learning, using different pretraining objectives. While these models achieve excellent results on the downstream NLP tasks, various domain adaptation techniques can improve their performance on domain-specific tasks. We compare and analyze the pretrained Neural Language Models, XLNet (autoregressive), and BERT (autoencoder) on the Legal Tasks. Results show that XLNet Model performs better on our Sequence Classification task of Legal Opinions Classification, whereas BERT produces better results on the NER task. We use domain-specific pretraining and additional legal vocabulary to adapt BERT Model further to the Legal Domain. We prepared multiple variants of the BERT Model, using both methods and their combination. Comparing our variants of the BERT Model, specializing in the Legal Domain, we conclude that both additional pretraining and vocabulary techniques enhance the BERT model's performance on the Legal Opinions Classification task. Additional legal vocabulary improves BERT's performance on the NER task. Combining the pretraining and vocabulary techniques further improves the final results. Our Legal-Vocab-BERT Model gives the best results on the Legal Opinions Task, outperforming the larger pretrained general Language Models, i.e., BERT-Base and XLNet-Base.

Acknowledgments

Firstly, thanks to God for blessing me with the abilities to successfully complete the thesis. Then, I wish to express my utmost gratitude to my thesis advisor Prof. Dr. Jelena Mitrović, for her continuous guidance throughout the research. She was always available to help me with her insightful suggestions and feedback on the research work. She is an excellent research advisor, provided me with great ideas, and kept me in the right direction to accomplish the thesis work.

I wish to thank Prof. Dr. Michael Granitzer for his valuable time for reviewing and evaluating my thesis as the second examiner. His suggestions at the kickoff presentation of my research helped speed up the research process.

I would like to thank the University of Passau for providing the computational resources to prepare the models for the research. I am grateful to the Harvard Law School Library for providing me with the legal data for research scholars from the CASELAW Access Project. I am also thankful to Google's Colaboratory Service for providing the computational resources used in the research.

Finally, I would like to acknowledge the love and support of my family and friends. Special thanks to my parents, who always believed in me; without their endless support and encouragement, this work would not have been possible.

List of Figures

List of Tables

1 Introduction

1.1 Motivation

In the last quarter of 2018, the field of Natural Language Processing was hit with a major paradigm shift after the release of Google's BERT (Bidirectional Encoder Representations from Transformers) [Dev+18], achieving state-of-the-art results on 11 NLP tasks. BERT introduced contextual representations learning in a bidirectional way, using masked language modeling techniques, which the previous efforts were lacking in capturing the contextual representations. A few months after BERT was released, another major development in the field was made by XLNet Model [Yan+19] released in June 2019, claiming to outperform BERT and achieving new state-of-the-art results on 20 NLP tasks. These new state-of-the-art Language models gained significant improvements by leveraging the power of deep learning techniques and the benefits of transfer learning. The major difference between BERT and XLNet is their objectives in learning the bidirectional context during the pretraining process, with BERT using autoencoder vs. XLNet using autoregressive language modeling.

In recent years, a significant increase of interest is seen in the application of NLP in the Legal Domain. Many researchers and tech companies are working on exploiting deep learning techniques, developing legal applications with NLP capabilities. In a field like Law, following a well-structured process with well-defined practices and detailed documentation makes it very ideal for applications of NLP. With NLP playing a vital role in law, and the new developments revolutionizing the field of NLP, it is very important for the Legal Domain and its community to understand better what new technologies are relevant for helping their cause and how would this impact the role that NLP is playing in the domain. Which of the new developments would fit their needs, and which techniques can enhance the performance of NLP tasks they require, and if the benefit gained is worth the effort needed to make use of these techniques.

1.2 Problem

Transformer [Vas+17] based language models like BERT and XLNet, use the techniques of unsupervised pretraining process on massive text to learn the features of languages and benefits of transfer learning to adapt to the downstream NLP tasks quickly by simple finetuning. Both Models (BERT and XLNet) provide advantages of transfer learning, easy finetuning, and faster computation in finetuning stage on the downstream NLP task, with the difference being the way each model learns the weights during the pretraining process. Deciding the model that fits the specific needs of natural language processing in the Legal Domain is an important part. It is helpful to have a meaningful comparison between the performance of two models with different pretraining strategies on the NLP tasks in the Legal Domain.

BERT uses self-supervised learning in the pretraining process and uses the WordPiece embedding vocabulary [Wu+16] which has around 30K tokens. This opens the path to improvement of the performance of the BERT Model on domain specific NLP tasks. This includes additional pretraining of the BERT Model on the legal text by initializing with the pretrained general BERT Model, as it is claimed to help improve performance on the domain-specific NLP tasks. The second option for enhancing BERT's performance on the Legal Domain is to add domain-specific, i.e., legal vocabulary in the existing WordPiece vocabulary. However, it is important to know how these methods may impact BERT's performance in accuracy and computational time. It is important to know when to opt for these options and if it is worth the effort and resources to use these domain adaptation techniques. These problems are the underlying foundation of the research questions described in Section-1.4.

1.3 Contribution

Following contributions are made that will help to solve the problems described in Section-1.2.

- A fair comparison of performance on NLP tasks in the Legal Domain between transformer models using different pretraining objectives to learn deep bidirectional context.

- Finding out the impacts of further pretraining BERT on Legal text. This includes the impact on the performance of accuracy as well as computation time during the finetuning stage.

- Finding out the impacts of adding legal vocabulary to WordPiece Vocabulary used by BERT on NLP tasks in the Legal Domain. This also includes both the performance of accuracy and computation time.

- Selecting the Legal vocabulary that would actually contribute to better results in the model's performance on Legal tasks.

- Providing the guidelines on exploiting the techniques to enhance performance in the BERT model. When is it appropriate to choose a performance enhancement technique, and what are the constraints that need to be taken care of before opting for a technique.

- We also release models prepared during the research, including Legal-BERT, Vocab-BERT, and Legal-Vocab-BERT models.

A meaningful comparison of new state-of-the-art transformer-based models will help select the model that fits the specific requirements of natural language processing in the Legal Domain. BERT Model provides excellent performance in natural language processing, with a strong understanding of the language using general text from different domains. However, with possibilities of further improvement within a domain, it will be very helpful to know which techniques would actually have a fruitful impact. Given that pretraining is an expensive procedure, it is not always easy to use the technique without knowing if going through these efforts would actually lead to the expectations.

1.4 Research Questions

1. **Which of the Transformer based Language Models such as BERT and XLNet perform better in Legal Domain?**

   Language Models are a vital part of the NLP pipeline. BERT and XLNet are the latest developments in this area, achieving state-of-the-art results in NLP tasks. Both models learn the deep bidirectional context during their pretraining stage and can be easily finetuned on downstream NLP tasks. Answering this question

will provide the findings on a comparison between performances of autoencoder models like BERT and autoregressive models like the XLNet.

2. **How does Legal Vocabulary affect the performance of BERT on Legal Tech?**

   For tokenization, BERT uses the vocabulary from WordPiece embedding, which contains around 30K tokens. The focus of this question is to find out the impacts of adding Legal Vocabulary, along with WordPiece vocabulary, to the pretrained BERT model. The impact includes the performance of BERT in evaluation measures as well as computation time on the downstream NLP tasks in the Legal Domain. The answer will provide guidelines on when to opt for this technique and how to select the vocabulary for a task.

3. **Is additional pretraining with Legal data on top of an existing pretrained general language model of BERT worthwhile for optimizing NLP performance in Legal Tech?**

   BERT is pretrained on a huge amount of plain text, covering the data from different domains and understands the general language features. However, as per the BERT's authors recommendation, it is beneficial to add domain-specific pretraining when working with NLP on a specific domain. This question focuses on the benefits gained by additional pretraining and the effort required to run the model's additional pretraining. It compares different variants of the BERT Model, including a BERT model pretrained on general plain text, a BERT Model with additional pretraining on Legal text, and a BERT Model combining the benefits gained by adding Legal vocabulary along with the additional pretraining on Legal text.

1.5 Structure of the Thesis

Thesis contains 8 Chapters in total. Chapter-1 introduces the research topic, including motivation, problem, and contribution and impact made by the Paper in this research area. The background knowledge, state-of-the-art, with necessary understanding of the models being studied, is explained in Chapter-2. Related work is discussed in Chapter-3. Chapter-4 contains the details on the preparation of Datasets and Models for the NLP

tasks performed. Experimental setup, used parameters, and implementation details of NLP tasks, are discussed in Chapter-5. Chapter-6 shows the results found from experimentation. A brief discussion on the findings and the rationality for the outcomes, is talked about in Chapter-7. Finally the conclusions and future work are discussed in Chapter-8.

2 Background

Natural Language Processing has been around for a few decades now. It evolved over the years with new techniques of understanding the natural language, development of different language models, from N-grams to Neural Language Models. Even though NLP, in terms of advancements, is a fairly young field, the last few years have seen a rapid progress. Recent developments in the field have made a huge impact on its efficiency, attracting the attention of researchers as well as firms working with NLP.

2.1 Natural Language Processing in Legal Domain

Law is a domain, which works with well-structured language and well-organized processes. To leverage the benefits of software and technology, the Legal Domain is dedicating efforts for its digital transformation to facilitate, simplify, and optimize legal processes with the development of Legal Tech[1]. Law has a formal language, with particular syntax, semantics, and its own specific vocabulary. As a matter of fact, Legal English is often referred to as a "sublanguage", and services like lawlinguists[2] are explicitly teaching it to the Lawyers and people working with the Legal Domain. These characteristics of Legal language make it a perfect subject for exploiting the Natural Language Processing's powers in the Legal Domain. Currently, NLP is playing a vital role in the Legal Technology in quite a few areas, including extracting relevant information about judicial decisions, automated review of contracts for error checking, automating the regular legal procedures and documentation, and Lawyer bots for Legal Advice.

---

[1]`https://en.wikipedia.org/wiki/Legal_technology`
[2]`https://lawlinguists.com/training/`

2.2 Background of Language Modeling

One of Natural Language Processing's fundamental tasks is predicting text conditioned on the previous sequence of words, which is the core of Language Modeling. The prominent statistical models were rule-based or probabilistic Language Models through the development of language modeling techniques, e.g., N-gram. The simplicity of a model like N-gram limits its performance over long texts. Neural Language Models helped to overcome these limitations and outperformed Statistical Language Models in language modeling tasks. Recurrent Neural Networks (RNNs) faced problems such as vanishing gradient, resulting in the inability to capture relevant dependencies in long sequences. Long Short Term Memory (LSTM) [Gra13] solved problems other RNNs faced with the capability of capturing long-term dependencies in sequences. The refinement of language modeling through Deep Learning methods continued up to the Transformer's [Vas+17] introduction, based one which the new state-of-the-art language models are built, including BERT and XLNet.

2.2.1 Context Learning

Word Representation is one of the main building blocks for Natural Language Processing, having a remarkable impact on performance, more specifically on Deep Learning models. There are different types and numerous techniques for creating word representations. Word representations can be fixed as generated by methods like "dictionary lookup" and "One-Hot Encoding" or distributed such as word embedding. Distributed Representations provide the key contribution to the efficiency of Deep Learning methods in NLP. A good word representation method should capture syntactic and semantic regularities of the natural language and incorporate the notion of word similarity.

Contextualized Word Representations, as compared to Static Word Representations, yield significant improvements in performance on NLP tasks. Same words can change their meanings depending on the context of the sentence, which is not comprehended by context-free representations. Context-free models will generate the same representations for a word, irrespective of the words it is surrounded by. For example, given two sentences, "Opening a bank account" and "Walking along the river bank", they will gen-

erate the same representation for the word "bank". Models like GloVe[3] and word2vec[4] generate context-free representations. Contextual representations incorporate the phenomena possessed by Homonym words, and represent them according to their context. Capturing context information with contextual representations provided remarkable improvement in language understanding and developing state-of-the-art language models. Linguist J.R Firth summarized this principle as "You shall know a word by the company it keeps" (J.R Firth 1957).

Learning Contextual representations can be done in a unidirectional or bidirectional way. OpenAI GPT [Rad+18] learns the context in left-to-right direction and generates pretrained representations only from the context before a word. ELMo [Pet+18] learns contextualized word representations in a bidirectional way by parsing the sentence in a left-to-right and right-to-left manner and then concatenating the learned representations from both directions, making it shallowly bidirectional. ULMFiT [HR18] learns forward and backward contexts and takes average predictions of both forward and backward models after independently finetuning for both. ELMo and ULMFit are limited to shallow bidirectional learning due to the sequence to sequence technique of underlying RNN architectures. The introduction of Transformer [Vas+17] enabled deep bidirectional learning, as it removes sequential dependency by reading all the words from a sentence at-once. BERT [Dev+18] exploits the Transformer's capability of reading sequences atonce to learn deep bidirectional context simultaneously, unlike the mechanism adopted by ELMo.

2.2.2 Neural Language Models

Conventional methods for language modeling used rule-based systems or probabilistic models, which had limitations of capturing contextual information in long sentences, and problems like running into "curse of dimensionality" as the vocabulary size increased. Deep Neural Network Models overcame these limitations with their complex and sophisticated architecture. Recurrent Neural Network shows improvement in capturing context information, but as input is fed serially, i.e., word by word, it makes them slower. In addition, RNNs face problems in maintaining the state over long sentences to capture the context. LSTM deals better with long sequences but is even slower than

---

[3]https://nlp.stanford.edu/projects/glove/
[4]https://www.tensorflow.org/tutorials/text/word2vec

other Recurrent Neural Networks. Transformer Neural Network solves these problems by a pure self-attention approach, where a complete sequence is passed to the model simultaneously. Contextual word representations are taken from Neural Network's hidden states by feeding the word embedding through Neural Network. This idea made a major impact on creating Deep Contextual Word Representations. Researchers scaled up the idea to huge neural networks, creating state-of-the-art language models like ELMo, GPT, BERT, and XLNet [Yan+19], by training on massive amount of unlabelled text. The study [Eth19] on deep contextual models shows that the upper layers of these models produce more context-specific representations.

2.2.3 Pretraining

The Pretraining process involves training a Neural Network, on a large amount of unlabelled text, in an unsupervised manner, using a language modeling objective to learn common language features, which can later be re-used to perform a supervised NLP task. In their Paper [DL15] on Semi-Supervised Sequence Learning, the authors presented two different approaches for using unlabelled data for training RNN. The first approach is the key task of a conventional language model, i.e., next-word prediction. The second approach is sequence autoencoding, where an input sequence is read into a vector and then predicted. Using these two techniques, a Neural Network can be pretrained to learn parameters in an unsupervised fashion, that can generalize well in the later supervised learning tasks. Pretraining is a one-time process, and weights learned can be used on many downstream NLP tasks, with simple finetuning, saving a lot of time and effort. This idea is an important technique that helped the current state-of-the-art models achieve better performance in many NLP tasks.

One of the major advantages of the pretraining process is the availability of training data. The unlabelled text is available in abundant amounts compared to scarcely available labeled data for specific tasks. OpenAI's GPT [Rad+18] combined the idea of unsupervised pretraining with Transformer Network [Vas+17] to gain state-of-the-art results on various NLP tasks. Results from GPT showed that the idea of pairing unsupervised pretraining with later supervised learning could yield better performance. The idea of unsupervised pretraining enables transfer learning for the pretrained models on NLP tasks, much like the idea of transfer learning that already exists in computer vision. Researchers exploited the idea in recent years, using different pretraining motives and

underlying Neural Network architectures, and trained on a large amount of unlabelled data to create the state-of-the-art pretrained language models. Some of the well-known pretrained models are GPT pretrained on 800M tokens, BERT pretrained on 3.3B tokens, XLNet pretrained on 32.89B tokens, and GPT-2 pretrained on 40B tokens of the plain text corpus.

2.3 Invention of Transformer

The introduction of Transformer in 2017 was a paradigm shift in the field of neural language modeling. Inspired by attention-mechanism [BCB14], in their paper "Attention Is All You Need" [Vas+17], the authors proposed a new Neural Network architecture solely based on the attention-mechanism. The transformer's architecture provides the ability to read-in a whole sequence at-once, removing the recurrence and convolution. This ability of the Transformer's architecture also enables parallelism and exploiting the power of the GPUs. Reading complete sequence at-once also paves the way for better context learning, which is well exploited by models like BERT, to learn left-to-right and right-to-left context simultaneously.

2.3.1 Transformer's Architecture

The Transformer is solely based on the Attention Mechanism instead of recursion and backpropagation. As shown in Figure-2.1, the transformer model has 2 parts, an encoder and a decoder block for performing sequence to sequence tasks. The encoder-decoder architecture is similar to the previous sequence to sequence models like LSTM but differs in the mechanism, which is self-attention instead of recursions. The encoder is given the vectorized input sequence, along with the position encoding. It encodes the input sequence, which is then used by the decoder to make the predictions for all the individual tokens of the output sequence. The decoder is given the output sequence that is so far generated, along with the positional encoding. It combines it with the encoded input sequence to predict the next possible tokens. As described in Section-2.3.3, Positional Encoding is the mechanism needed to keep the order in which the sequences are fed into the transformer model.

Figure 2.1: Transformer's Architecture. From [Vas+17]

Self-Attention Mechanism is explained in Section-2.3.2, which is used as a sub-layer by both Encoder and Decoder of the Transformer Model. As shown on the left side of the Figure-2.1, the Encoder is stacked with multiple layers (where Nx represents the number of layers). Each layer contains a Multi-Head Self-Attention sub-layer, followed by a fully connected feed-forward network. Both sub-layers are followed by a deep residual learning [He+15] part and normalization. As shown on the right side of the Figure-2.1, the Decoder is a stack of multiple layers, represented by Nx. Each decoder layer has one additional multi-head attention sub-layer along with the two sub-layers used by the encoder, which computes attention over the output it receives from the encoder. Applying residual connections and normalization is done by each individual sub-layer of the decoder. The decoder takes the output sequence predicted so far as input, along with the encoded input sequence from the encoder; hence the Multi-Head Attention sub-layer of the decoder is masked to prevent attending to the outputs that are not yet predicted in the output sequence. The decoder generates Output Probabilities for the next tokens; after applying the $Softmax$ function, and the highest probability is selected as the next token predicted by the Transformer Model in the output sequence.

2.3.2 Self-Attention Mechanism

Authors of transformer argue that using the traditional RNN's recurrence over hidden states of the long input sequence results in loss of information over the long sequences. They propose to use the attention mechanism while predicting any particular output token instead of going through the recurrence mechanism. The attention mechanism makes the computation much faster as it excludes the need for multi-step backpropagation, as in RNN. The complete input sequence is fed to the encoder, and the decoder generates the probability of the next word of the output sequence in just one iteration. Getting rid of recurrence adds more parallelism and helps utilize the parallel computation power of GPUs.

Attention essentially means that the model should be able to figure out which words in the input sequence must be paid attention to when making a particular prediction. The decoder follows an addressing scheme to decide where to look at, in the input sequence. The encoder generates the Keys (K) and Values (V) from the encoded input sequence. The first Multi-Head Attention sub-layer of the Decoder generates Queries (Q) from the

Scaled Dot-Product Attention

Multi-Head Attention

Figure 2.2: Transformer's Scaled Dot-Product Attention and Multi-Head Attention. From [Vas+17]

output sequence generated so far. Keys, Values, and Queries are fed into the second Multi-Head Attention sub-layer of the Decoder.

Figure-2.2 shows the Scaled Dot-Product Attention on the left and the Multi-Head Attention on the right side. The transformer uses one of the most commonly used attention mechanisms, i.e., Dot-Product Attention, and adds a scaling factor before taking softmax; hence the name "Scaled Dot-Product Attention". The authors selected Dot-Product due to its efficiency in time and memory consumption. Equation-2.1 [Vas+17] shows the computation of the Scaled Dot Production Attention mechanism. A dot-product of Query (Q) is computed with all the Keys (K), both Q and K of dimensions d_k. A factor of $\frac{1}{\sqrt{d_k}}$ scales the result to avoid running into vanishing gradient problem when applying softmax to relatively large results from the dot product of Q and K. The result of the dot-product between Keys and Query is masked to avoid demolishing the autoregression by allowing each position to look only at the output generated so far by the decoder. The dot product yields maximum value when two vectors are fully aligned, and the dot product tends to get smaller with the increase of angle between the vectors. The dot-product yields higher results for the Key and Query vectors which are most aligned, and then softmax pushes the gap between the results from the dot products

of different Keys and Query vectors, and the Key that yields the largest dot product with the Query is selected, which identifies the part which model should pay attention to when predicting next output token. The next token is selected by multiplying the result from softmax with the Values vector.

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \qquad (2.1)$$

Multi-Head Attention is basically parallel Dot-Product Attentions, as shown in Figure-2.2, where h is the number of linear projections for Keys, Values, and Queries. These multiple attentions can be computed in parallel, and their results are concatenated to produce final results. Multiple attentions can learn different representations of Keys, Values, and Queries. In terms of language understanding, an attention-head can be considered as paying attention to a specific part of the language structure and semantic, e.g., personal pronouns. The modern language models use different combinations of these Multi-Head Attentions to achieve better performance on the Natural Language Processing tasks.

2.3.3 Positional-Encoding

Figure-2.1 shows that both encoder and decoder of the Transformer Model take Positional Encoding along with Input Embedding and Output Embedding, respectively. Since Transformer Model is purely based on the Attention Mechanism, unlike a recurrent network, there is no concept of time-step in the Transformer model; hence it cannot remember the order in which sequence was fed into the model. Positional Encoding is a mechanism to maintain the order of input since the Transformer is reading the whole sequence at-once. Relative positions of tokens in the input and output sequences are given to the encoder and decoder. Positional Encoding is the vector with information about distances between the words in the input/output sequence. After applying positional encoding on the input embedding, the word vector is generated with the positional information, i.e., context, so the positional encoding gives the notion of word context in the sentence. The Transformer [Vas+17] uses a sin-cosine function to generate the Positional Encoding vector. The dimension of the positional encoding vector is the same as the dimension of input/output embedding, i.e., $d_{word\_embedding} = d_{positional\_encoding}$. The

positional embedding vector is added into the input/output embedding, which injects the order of words without integrating into the Transformer model itself.

2.4 BERT Model

BERT (short for Bidirectional Encoder Representations from Transformers) [Dev+18] is a revolutionary deep learning language model introduced in the last quarter of 2018. BERT uses representations learned in a bidirectional way by encoder part of the Transformer Neural Network, hence the name. BERT combines the strengths of the latest Language Modeling trends to produce state-of-the-art results on various Natural Language Processing tasks. It uses the recent Neural Network, i.e., Transformer [Vas+17], as its underlying architecture. BERT also takes advantage of Transfer Learning by Pretraining on the plain text in an unsupervised manner. The learning gained in the Pretraining process can later be used by simple Finetuning on downstream Natural Language Processing tasks. Transformer Neural Network enables BERT to learn deep bidirectional context. Bidirectional learning enables BERT to represent each word in a sequence according to its previous and next contexts, in contrast to unidirectional learning methods, which encode a word using only either previous or next context, depending on the direction in which they learned the context. GPT [Rad+18] from OpenAI is a unidirectional Transformer Model. ELMo [Pet+18] concatenates LSTM trained in left-to-right and right-to-left separately, making it shallowly bidirectional. BERT, on the other hand, learns the bidirectional context simultaneously.

Section-2.4.2 explains the pretraining process of BERT and how it achieved deep bidirectional context learning using the masked language modeling technique. BERT was pretrained on plain text from BookCorpus [Zhu+15] comprising 800 million words and text passages extracted from English Wikipedia, including 2500 million words. Pretraining is an expensive but one-time process. On cloud TPUs (Tensor Processing Units), pretraining BERT can take around 4 days. Authors released multiple variants for BERT Model, available on googleapis' bucket[5]. Pretrained BERT Models are now available for different natural languages, including a Multilingual BERT Model[6], which supports more than 100 natural languages.

---

[5]`https://storage.googleapis.com/bert_models/2020_02_20/all_bert_models.zip`
[6]`https://github.com/google-research/bert/blob/master/multilingual.md`

Figure 2.3: Pretraining architecture of BERT. From [Dev+18]

2.4.1 BERT's Architecture

The encoder of the Transformer Neural Network explained in Section-2.3.1, is used as the building block by BERT in its underlying architecture. BERT is a stack of multiple layers of Transformer Encoders. The paper [Dev+18] introduced two different configurations of pretrained BERT Models, i.e., **BERT$_{\textbf{BASE}}$** (L=12, H=768, A=12) and **BERT$_{\textbf{LARGE}}$** (L=24, H=1024, A=16), where L represents the number of layers, i.e., Transformer blocks, H is the size of hidden states, and A is the number of Attention Heads. The number of parameters learned by **BERT$_{\textbf{BASE}}$** and **BERT$_{\textbf{LARGE}}$** are 110M and 340M, respectively.

Figure-2.3 shows the flow of information of words in the pretraining architecture of the BERT Model. Starting from the bottom layer of input embeddings (E_1, E_1, ..., E_N), information flows through all the encoder layers, with intermediate states (Trm, Trm, ..., Trm) until the final encoded output (T_1, T_1, ..., T_N). Intermediate states of the same size are generated at each layer by computing multi-headed attention on the previous layer's word representation. The direction of arrows in Figure-2.3 shows that

each word representation is learned by looking at its previous and next context.

2.4.2 Pretraining BERT

The pretraining process is explained in Section-2.2.3. It involves unsupervised training of a Neural Network on a huge amount of plain text, with some pretraining Objectives. A Large Model with multiple Transformer Encoder layers, as explained in Section-2.4.1, is trained on a large corpus of plain text, using one million training steps, for a long time, and the outcome is the BERT Model. Pretraining BERT involves two main pretraining Objectives, i.e., **Masked Language Modeling** and **Next Sentence Prediction** explained in Section-2.4.3 and Section-2.4.4, respectively.

Pretraining BERT Model is a reasonably expensive task. English BERT is pretrained on the plain English text of 3.3 Billion tokens. Using the pretrained BERT Model released by the authors is suitable in most of the cases. However, further pretraining BERT with domain-specific corpora can help increase the performance in domain-specific tasks, as compared to general BERT model, which is pretrained on text from different domains.

BERT's authors provide the implementation for running Masked Language Modeling and Next Sentence Prediction tasks on arbitrary text to run pretraining on a BERT Model of our choice. Pretraining process involves two steps:

1. **Create Pretraining Data[7]**

   In the first step, data from plain text files are converted into TFRecord[8] file format for BERT.

2. **Run Pretraining[9]**

   In the second step, BERT is pretrained on Masked Language Modeling and Next Sentence Prediction tasks using the TFRecord data generated in the first step. Refer to BERT's repository[10] for technical details on input parameters and pretraining recommendations from authors.

---

[7]`https://github.com/google-research/bert/blob/master/create_pretraining_data.py`
[8]TFRecord is a simple format from TensorFlow for storing sequence of binary records
[9]`https://github.com/google-research/bert/blob/master/run_pretraining.py`
[10]`https://github.com/google-research/bert#pre-training-with-bert`

2.4.3 Masked Language Modeling

BERT takes the idea of Masked Language Modeling (MLM) from the Cloze task [Tay53]. It involves randomly masking some tokens from the input and then predict the masked words based on their context. MLM is the intuition behind the deep bidirectional context leaning of the BERT Model. The traditional language modeling is done in a unidirectional way, either left-to-right or right-to-left at a time. Masking the input helps to achieve it from both directions simultaneously without revealing the word to be predicted. BERT randomly replaces 15% of the input tokens by `[MASK]` for prediction to learn enough context, yet avoiding overly expensive training. Given the example below, BERT aims to predict the masked word in the sentence by looking at their surrounding context.

Input Sentence: Animal abuse is a crime, and it is punishable by law.

Masked Input Sentence: Animal abuse is a `[MASK]` , and it is punishable by `[MASK]` .

The masked language modeling objective's problem is that masked tokens during the pretraining stage are never seen in the later finetuning stage. BERT makes the masked data biased to solve this problem. To make the data biased, when masking randomly selected 15% input tokens, 80% are actually replaced by `[MASK]`, 10% are replaced with a random word, and the remaining 10% are retained.

2.4.4 Next Sentence Prediction

The Next Sentence Prediction task involves pretraining sentence-pair representations to learn relationships between sentences. Given two sentences, A and B, with A appearing before B, the task is to predict if the sentence B actually follows sentence A or is it a randomly selected sentence from the Corpus. The input corpus of plain text have a defined structure, the input plain text file must have one sentence per line, and sentences must be in the correct order. Both pretraining objectives require a plain text format that is simple and easy to generate from plain text documents. Input representation of both pretraining objectives is explained in Section-2.4.6.

2.4.5 WordPiece Vocabulary and Tokenization

BERT uses WordPiece [Wu+16] Vocabulary and Tokenization to generate embeddings. WordPiece is a subword segmentation algorithm, and its vocabulary contains around 30,000 tokens. A missing input word from vocabulary is greedily broken down into subwords until all the sub-words are found in the vocabulary to handle the out-of-vocabulary problem. The authors provided the implementation[11] of end-to-end tokenization and encoding to generate the input embeddings. Below is an example of Tokenization and Encoding of arbitrary legal text using BERT's WordPiece Tokenization.

Input Text: The Court had erred in overturning original decision

Tokenized Input: the court had er ##red in over ##turn ##ing original decision

Encoded Input: 101 2006 13474 1005 1055 4367 1010 1996 2212 2457 7219 1996 12087 1012 102

where the first token in the Encoded Input, i.e., 101, represents the special token `[CLS]`, which BERT adds as the first token of a sequence. Ending Token 102 represents the special token `[SEP]`, added by BERT, to separate two sentences from a sentence-pair. These tokens are added when the input sequence is encoded with the BERT model.

2.4.6 BERT's Input Representation

BERT provides a single yet unambiguous input representation for both single sentence and sentence-pair tasks, hence easily adapting to a variety of NLP tasks. Figure-2.4 shows the input representation of BERT Model. The maximum length of the input sequence can be up to 512 tokens. Token Embeddings, taken from the input text, Segment Embeddings distinguishing the sentences from sentence-pair, and Positional Encoding representing the context of words in the input sequence, add up to generate Input Embeddings. A single sentence input sequence ends with a special token `[SEP]`. Sentences are separated by the special character `[SEP]` in a sentence-pair. During the pretraining step, 15% of the tokens are masked from the input sequence, hence the input representation satisfying the requirements of both pretraining Objectives, i.e., Masked

---

[11]`https://github.com/google-research/bert/blob/master/tokenization.py`

Figure 2.4: Input Representation of BERT. From [Dev+18]

Language Modeling, and Next Sentence Prediction. During the finetuning step, the input representation uses the final hidden state of the first token, i.e., `[CLS]`, for aggregate sequence representation of classification tasks. Similarly, it fits to the need for sentence-pair tasks such as Language Inference in the finetuning stage. The ingenuity of BERT's input representation design makes it portable between pretraining and finetuning stages, on various tasks, without any significant task-specific changes.

2.4.7 Finetuning BERT

The pretrained BERT model can be easily finetuned on various downstream NLP tasks. Finetuning is comparatively much less expensive and can be done in a matter of a few minutes to a couple of hours, depending on available resources and tasks being performed. Authors recommended best finetuning parameters for a variety of downstream NLP tasks, trained for 4 epochs:

- batch sizes: 8, 16, 32, 64, 128

- learning rates: 3e-4, 1e-4, 5e-5, 3e-5

All the parameters are tuned in the Finetuning stage with the same learning rate selected for a task. The top-right part of Figure-2.5 shows finetuning configurations for a sequence classification task. Input Representations, as discussed in Section-2.4.6, are easily adapted to the sequence classification task. Pretrained BERT layers are trained simultaneously on the input embedding fed to give an output representation. The hidden state of the special token at the beginning of the input sequence, i.e., `[CLS]`, is fed to the classification layer as aggregated sequence representation. The classification

Figure 2.5: Finetuning BERT on different Tasks. From [Dev+18]

layer is added in the finetuning stage. It has a dimension of **LxH**, where L is the number of classes/labels of the sequence classification data, and H is the hidden size of the pretrained BERT model. $Softmax$ is applied at the end to compute probabilities for output labels.

For sentence-pair classification tasks, such as Natural Language Inference, the input representation is quite similar. The only change compared to the single sentence classification task is that two sentences are packed together in the input sequence, using a special token, i.e., [SEP], to separate both sentences, as shown in the top-left part of Figure-2.5. Special token [CLS] is used for feeding aggregated sequence representation to the classification layer, similar to the single sentence classification task.

The finetuning setting for a token classification task such as Named Entity Recognition is shown on the bottom-right part of Figure-2.5. In token classification, every token's

final hidden output is fed to the classification layer instead of just the first token, i.e., `[CLS]` as in the case of the sequence classification task, and a label is predicted for every token. For tokens, broken down by WordPiece tokenizer into multiple parts, the prediction is made only for the first part.

The bottom-left part of the Figure-2.5 shows the finetuning setting for the Question-Answering task. In the Question-Answering task, a question and a context paragraph containing the answer to the question is given. The model predicts the beginning and ending tokens from the paragraph that possibly mark the span of the answer in the paragraph. Like the sentence-pair classification task, the Question and the context paragraph are packed together in the input sequence, with a special token, i.e., `[SEP]` separating both from each other. Two new parameters, i.e., "start vector" and "end vector", are learned in this task during the finetuning. Final probabilities for starting and ending tokens are computed by taking softmax over each output token's product with "start vector" and "end vector", respectively.

2.5 XLNet Model

XLNet [Yan+19] is an autoregressive pretraining method that enables bidirectional context learning using Permutation Language Modeling as a pretraining Objective. Like BERT [Dev+18], XLNet also uses Transformer [Vas+17] as its underlying Neural Network for its training. Modern Neural Language Models use pretraining in their first stage to train themselves on a large amount of plain text of a natural language, in an unsupervised way. Two of the popular pretraining objectives used are autoencoding and autoregressive (AR) language modeling. BERT uses autoencoding-based pretraining, whereas XLNet uses autoregressive language modeling. Authors of XLNet argue that using autoencoding-based pretraining, BERT corrupts its input by masking[12] it during the pretraining stage. Special tokens, masking the words from input in the pretraining stage, are not part of real data during the finetuning stage on the downstream NLP tasks, making both stages inconsistent, resulting in the pretrain-finetune discrepancy problem.

The two most successful pretraining objectives, i.e., autoencoding and autoregressive language modeling, have their own pro and cons. BERT is based on denoising autoen-

---

[12]Masked Language Modeling, as explained in Section-2.4.3

coding, which maximizes the likelihood of a masked token x_t from input sequence x = $[x_0, x_1, \ldots, x_t, \ldots, x_T]$ based on all the tokens from input sequence x except x_t itself. Autoencoding (AE) enables BERT learning bidirectional context, but input masking leads to the pretrain-finetune discrepancy. Also, it assumes independence between masked inputs, which does not always hold in natural languages. Autoregressive (AR) language modeling aims to predict a token x_t based on the tokens that appear before it, i.e., $[x_0, x_1, \ldots, x_{t-1}]$ or after it, i.e., $[x_{t+1}, x_{t+2}, \ldots, x_T]$, depending on the direction. AR language Modeling does not rely on corrupted input, but it can only encode unidirectional context, in either forward or backward direction, lacking simultaneous bidirectional context learning. XLNet exploits the strength of both these objectives while avoiding their limitations using its ingenious Permutation Language Modeling pretraining objective. Using Permutation Language Modeling, XLNet overcame the gap between autoregressive language modeling and effective bidirectional pretraining. By combining the benefits of both types of objectives and avoiding their drawbacks, XLNet became the new state-of-the-art Language Model, outperforming BERT on 20 Natural Language Processing tasks.

2.5.1 Transformer-XL

The novel Transformer Neural Network Model [Vas+17], has a lot of potential, and helps to learn long term dependencies, enabling simultaneous bidirectional context learning. However, it has a certain limitation on the length of the input sequence. In the beginning of the year 2019, Transformer-XL [Dai+19] overcame this limitation and enabled the attention mechanism beyond the fixed-length context.

Transformer-XL can learn long-term dependencies that are 80% longer than Recurrent Neural Networks and 450% longer than the Transformer. It achieves better performance on both long and short sequences in both evaluation measures and computation time. Transformer-XL achieves this using a new positional encoding scheme combined with a segment-level recurrence mechanism. State-of-the-art Language Model XLNet incorporates the ideas from Transformer-XL and is named after it.

2.5.2 XLNet's Architecture

The major contribution from XLNet is not its architecture, rather a superior pretraining method that adapts the benefits of different pretraining objectives. It borrows the ideas of segment recurrence mechanism and the novel relative encoding scheme proposed by Transformer-XL [Dai+19], as explained in Section-2.5.1. Underlying XLNet architecture is based on multiple Transformer layers, with recurrence. Multiple variants of different sizes and configurations of pretrained XLNet Models are released. The pretrained XL-Net Models include **XLNet$_{BASE}$** (L=12, H=768, A=12) and **XLNet$_{LARGE}$** (L=24, H=1024, A=16) where L represents the number of layers, i.e., Transformer blocks, H is the size of hidden states, and A is the number of Attention Heads. The number of parameters learned by **XLNet$_{BASE}$** and **XLNet$_{LARGE}$** are ∼110M and ∼340M, respectively. The size and configurations of both XLNet Models are comparable to BERT, although **XLNet$_{LARGE}$** is pretrained on much larger pretraining data than **BERT$_{LARGE}$**. However, to give a fair comparison, authors of XLNet also released another variant of **XLNet$_{LARGE}$**, namely **XLNet-Large-wikibooks**, which is pretrained on the same data, i.e., WikiPedia+BooksCorpus, like **BERT$_{LARGE}$**.

2.5.3 Pretraining XLNet

Pretraining data for XLNet includes BooksCorpus [Zhu+15] and text from English Wikipedia, on which BERT was pretrained. In addition, it also includes Giga5 [Par+11], ClueWeb 2012-B (an extension of [Cal+09]), and Common Crawl [19]. XLNet uses SentencePiece [KR18] tokenization on the input pretraining data after excluding the low-quality documents from raw corpora. The total number of tokens (after applying SentencePiece tokenization) included in pretraining data for XLNet are 32.89B, including Wikipedia (2.78B), BooksCorpus (1.09B), Giga5 (4.75B), ClueWeb (4.30B), and Common Crawl (19.97B). Largest XLNet Model, i.e., **XLNet$_{LARGE}$**, reuses all pretraining hyper-parameters as in **BERT$_{LARGE}$** and pretrains on Wikipedia+BookCorpus, making it equal in size and configuration. This model is further augmented by pretraining on all the remaining pretraining datasets. Pretraining XLNet Model took five and a half days on 512 TPU v3 chips for 500K training steps, using sequence length as 512 tokens and batch size as 8192. The finetuning process for XLNet is fairly similar to that of BERT.

Figure 2.6: Permutation Language Modeling for predicting x_3 from the different permutations of the same input sequence. From [Yan+19]

2.5.4 Permutation Language Modeling

Traditional Autoregressive Language Modeling cannot learn simultaneous bidirectional context. To use the advantages of AR Language Modeling, along with the deep bidirectional context learning, XLNet [Yan+19] uses the novel idea of Permutation Language Modeling, inspired by NADE [Uri+16], as its pretraining Objective. XLNet aims to improve pretraining using Permutation LM to predict all tokens in random order. The basic idea behind Permutation LM is Permutations. Given a sequence x containing T number of tokens, there are T! possible factorization orders, e.g., x = $[x_1, x_2, x_3]$, then 3! i.e., 6 possible permutations for the sequence are $\{[x_1, x_2, x_3], [x_1, x_3, x_2], [x_2, x_1, x_3], [x_2, x_3, x_1], [x_3, x_1, x_2], [x_3, x_2, x_1]\}$. An illustration of Permutation Language Modeling is shown in Figure-2.6. XLNet shares the parameters across all the permuted factorization orders to learn the context from all positions for a word in the sequence. It aims to maximize the log-likelihood of a sequence from the possible permutations of the factorization order, to capture the context information from both directions simultaneously. The mathematical formulation given by authors for the Permutation Language

Figure 2.7: Two-Stream Self-Attention. From [Yan+19]

Modeling is shown in Equation-2.2 [Yan+19].

$$\max_{\theta} \quad \mathbb{E}_{\mathbf{z} \sim \mathcal{Z}_T} \left[\sum_{t=1}^{T} \log p_\theta \left(x_{z_t} \mid \mathbf{x}_{\mathbf{z}_{<t}} \right) \right]. \tag{2.2}$$

where $\mathcal{Z}_T$ is a set of all possible permutations of the sequence of length T, x_{z_t} is the tth element, and $\mathbf{x}_{\mathbf{z}_{<t}}$ represents the tokens before it, in the permutation $\mathbf{z} \in \mathcal{Z}_T$ of the input sequence.

2.5.5 Two-Stream Self-Attention

XLNet reparameterizes the Transformer-XL architecture to use the Permutation Language Modeling objective. This is necessary in order to make the objective work with traditional Transformer/Transformer-XL architecture, to remove the target ambiguity. The Transformer-XL [Dai+19] looks at the entire input sequence simultaneously, and XLNet does not mask any tokens like BERT [Dev+18]. Hence it modifies the Transformer-XL to only look at the hidden representation of the tokens preceding the target to be predicted and the positional information of the target itself. To maintain the information about the order of input sequence, positional information is embedded using relative encoding.

Two-Stream Self-Attention involves two types of attention. The first one, content stream representation, is the same as the standard self-attention in the Transformer architecture.

It examines both content and the positional information of the token to be predicted. The second attention, query stream representation, replaces the input masking, i.e., `[MASK]`, as explained in Section-2.4.3. It learns to predict the target token using context and the target's positional information, but not the content of the target. If token x_i is to be predicted, the subsequent layers can only access its positional information and the content of tokens preceding it, but not the content of x_i itself. The attention is computed by changing the Query (Q), Key (K), and Value (V) vectors according to this modification. Figure-2.7 shows the content stream and query stream self-attentions.

3 Related Work

3.1 Evaluation of Deep Neural Language Models on Hate Speech Detection

Capturing long-term dependencies and a bidirectional context in the input sequence are key factors for improvements in deep neural language modeling. In their paper [MBG19] on detecting hate speech using Deep Learning Language Models, authors combined a Convolutional Neural Network (CNN) and bidirectional GRU (2 concatenated GRUs, trained in each direction, i.e., left-to-right and right-to-left) to build the proposed C-BiGRU Model, capturing long-term dependencies to distinguish offensive tweets from non-offensive tweets. C-BiGRU was used in another paper [Hus+20] participating in Se-mEval 2020 workshop, taking part in English, Turkish and Danish competitions, achieving F1 scores of 90.88%, 76.76%, and 76.70%, respectively.

3.2 Legal Area Classification

In a comparative study [HKC19] of different Text Classifiers on Judgments from the Supreme Court of Singapore, authors applied different machine learning approaches to classify the judgments in different legal areas. It gives a comparison between state-of-the-art Neural Language Models and Traditional Statistical Models. Their experiments tested different models, including Topic Models, Word Embedding Feature Models, and Pretrained Language Models. The authors concluded that while data limitation affects all classifiers, the state-of-the-art Neural Language Models like BERT are much less sensitive to data scarcity and can perform well on as few legal judgments as 588.

The study compares different types of models on legal judgment classification. In contrast to their work, we examine two different Neural Language Models in the Legal Domain, i.e., BERT [Dev+18], and XLNet [Yan+19], pretrained with different Pretraining Objectives, i.e., autoencoding and regressive language modeling, respectively. At the time of writing this paper, there is no comparison between these models already available on legal judgment classification.

3.3 German Legal Decision Corpora

A recent paper on Design and Implementation of German Legal Decision Corpora [UMG21] presents two German legal text corpora and experiments on these corpora. One of the corpora consists of German courts' decisions, whereas the second one is a subset of the first corpus, with the conclusion, definition, and subsumption components of German legal writing style, annotated by a domain expert. Based on the first corpus, the paper provides an experiment on the classification of decision type. The corpus contains decisions of various types, including types of resolutions, and judgments, etc., with imbalanced decision classes. To create balance, authors categorized all types of resolutions into a class *resolution*, and all types of judgments into a class *judgment*, and the remaining are labeled as *others*. They illustrated the corpus's usefulness by conducting a comparison between Logistic Regression (LR) and Support Vector Classification (SVC), in which SVC slightly outperformed LR. The second corpus is labeled with the components of Urteilsstil (judgment style). The paper shows results from experiments performed using LR and SVC, with unigram and tf-idf features. The goal is to train a model on Urteilsstil (judgment style) and use it on Gutachtenstil (appraisal style) to prove that the model can withstand a domain transfer, for detecting the common components between the two legal writing styles. All the combinations of classifiers and features outperformed the baseline, which is created by a decision stump that always predicts the majority class, showing that meaningful features are selected for domain transfer.

In contrast, to conduct experiments on Natural Language Processing in the Legal domain, we evaluate Deep Neural Language Models using US courts' decisions.

There has been a lot of research done on performance evaluation of BERT Model [Dev+18] on domain-specific corpora, including BioBERT Model [Lee+19] pretrained in

the biomedical domain, SciBERT Model [BCL19] pretrained on scientific datasets, and LEGAL-BERT Model [Cha+20] pretrained on Legal Data. Besides the domain-specific adaptation, BERT is also retrained by caselli et al. to create a language variety-oriented model, i.e., HateBERT [Cas+21].

3.4 HateBERT

In their paper on Retraining BERT for Abusive Language Detection in English, authors released HateBERT [Cas+21], an abuse-inclined version of the BERT model. HateBERT is prepared by further pretraining the $BERT_{base-uncased}$ model on a total of 43,379,350 tokens from the RAL-E (Reddit Abusive Language English) dataset. The RAL-E dataset is also part of the contribution, which is prepared by extracting a publicly available Reddit Comments collection from communities banned for hosting hateful, offensive, and abusive content. Authors performed experiments with multiple datasets, including OffensEval 2019 [Zam+19], AbusEval [Cas+20], and HatEval [Bas+19]. HateBERT outperforms the general BERT model on all the datasets and is more robust in improvements on negative and positive classes from hate speech data. Authors conclude that further pretraining is an effective technique to adapt BERT to other language varieties. HateBERT obtains robust representations for the offensive phenomenon against which it is finetuned. It adapts better to the language variety and polarity, as compared to its corresponding general BERT model.

In comparison, instead of language variety, we choose the domain of Law for evaluation of domain adaptation abilities of the BERT Model.

3.5 BioBERT

To develop BioBERT [Lee+19], authors used data from archives of biomedical and life science literature, comprising a Corpus 18B words in total, including 4.5B words from PubMed[1] and 13.5B words Corpus from PMC[2]. BioBERT is an outcome of their investigation on how the BERT model can be adapted to the medical domain. The authors

---

[1]`https://catalog.data.gov/dataset/pubmed`
[2]`https://www.ncbi.nlm.nih.gov/pmc/`

built the model BioBERT by initializing it with the weights learned from pretrained BERT$_{\text{BASE}}$ Model. Authors released a model trained on PubMed and PMC corpora, using 470K training steps, which until this release was their best performing model. Later, they released another model, which was trained only on PubMed corpus using 1M training steps, which performed better than their first model. Authors found that BioBERT largely outperforms BERT in biomedical text mining tasks.

In comparison to their work in the biomedical domain, we propose a Legal-BERT Model specializing in the Legal Domain. In addition to pretraining a general pretrained BERT model on legal corpora, we also explore the benefits of adding Legal vocabulary in the general WordPiece vocabulary used by BERT, comprising around 30K tokens. We initialized our Legal-BERT Model with pretrained BERT$_{\text{MEDIUM}}$ Model and pretrained it further for 1M training steps on legal data. We compare our Legal-BERT Model's performance with the general BERT Model in the Legal Domain to analyze the effects of Legal Domain adaptation.

3.6 SciBERT

SciBERT [BCL19] is a pretrained BERT Model on a large amount of scientific text. It is pretrained from scratch, using the same configuration and size as BERT$_{\text{BASE}}$ instead of initializing it with weights of an existing pretrained BERT Model. The pretraining data is taken from Semantic Scholar Corpus[3], which includes Biomedical (82%) and Computer Science (18%) papers building a corpus of 3.17B tokens. Along with the SciBERT Model, authors also constructed a SCIVOCAB, i.e., the WordPiece vocabulary generated from their scientific corpus. Keeping the size of SCIVOCAB the same as BERT$_{\text{BASE}}$, authors compared the evaluation of SciBERT and BERT$_{\text{BASE}}$. They concluded that SciBERT significantly outperformed BERT$_{\text{BASE}}$ in the scientific domain. They performed several NLP tasks, ranging in Computer Science, Biomedical, and multidomain tasks, where SciBERT achieved state-of-the-art results in 8 out of 12 tasks.

In comparison to SciBERT, our Legal-BERT Model, as discussed in Section-4.2.1, specializes in Legal Domain. There are two major differences besides the domain expertise between SciBERT and Legal-BERT. The first difference is the BERT Model's size, i.e., SciBERT has the size and configurations of BERT$_{\text{BASE}}$, whereas our Legal-BERT has

---

[3]https://www.semanticscholar.org/

BERT$_{\text{MEDIUM}}$. To analyze the effects on performance, Legal-BERT is compared with pretrained general BERT$_{\text{MEDIUM}}$ as released by the BERT's authors. The second difference is that SciBERT was pretrained from scratch, whereas our Legal-BERT is initialized with BERT$_{\text{MEDIUM}}$ and then further pretrained on Legal text.

The difference between SCIVOCAB and our work in analyzing the effects of Legal Vocabulary on the BERT Model is that SCIVOCAB was constructed from the scientific corpus, overlapping original WordPiece Vocabulary[4] by 42%. In contrast, we add new Vocabulary in the original WordPiece vocabulary, which is specific to the Legal Domain, i.e., legal terminologies. The newly added Legal Vocabulary is not broken down into multiple parts, contrary to WordPiece Vocabulary created using SentencePiece[5] library. This increases the size of the original WordPiece vocabulary, and we analyze the effects on performance in evaluation measures and computation time on legal tasks.

3.7 LEGAL-BERT: The Muppets straight out of Law School

In the last quarter of 2020, a LEGAL-BERT Model was released by researchers in their paper [Cha+20] during the writing of our paper. The authors developed a family of Legal BERT Models. They prepared a pretrained BERT Model from scratch on Legal text (named LEGAL-BERT-SC), with configurations and size of the BERT$_{\text{BASE}}$ model. The authors initialized the second Legal BERT Model (named LEGAL-BERT-FP) with pretrained BERT$_{\text{BASE}}$ and added further pretraining on legal text for 1M training steps. Authors also released a Legal BERT Model of smaller configuration and size, i.e., LEGAL-BERT-SMALL with 6 layers, 512 hidden units, and 8 attention heads. Authors collected legal text of 12GB from multiple legal sub-domains for pretraining Legal BERT Models. Pretraining data collected contained legal documents from EU Legislation (16.5%), UK Legislation (12.2%), European Court of Justice (ECJ) cases (5.2%), European Court of Human Rights (ECHR) cases (4.3%), US court cases, from Case Law Project (27.8%), and US Contracts (34.0%). Within the Further Pretrained Legal BERT Models, i.e., LEGAL-BERT-FP, authors developed legal models specializing

---

[4]the original WordPiece vocabulary that comes with pretrained BERT Model, constructed from general multi-domain English text

[5]`https://github.com/google/sentencepiece`

in legal sub-fields, e.g., US Contracts, ECJ, ECHR. The authors concluded that a Legal variant of BERT always outperformed pretrained $BERT_{BASE}$ on all legal datasets. The LEGAL-BERT-SMALL compares well to Legal variants of BERT Models, despite being small in size. Within the LEGAL-BERT-FP family, sub-domain specific models adapt faster in sub-domain tasks than a variant pretrained on the complete collection of legal corpora.

The fundamental differences between Legal BERT released by authors of the paper "LEGAL-BERT: The Muppets straight out of Law School" [Cha+20], and our Legal-BERT Model, are the size and configurations, and the pretraining data. Our Model is pretrained on Legal text from both publicly accessible as well as Research Scholar's data of US Cases from the Case Law Project, whereas they used legal text from various subfields within the Legal Domain from publicly accessible resources. Our Legal Model has the size and configurations of a relatively smaller BERT model, i.e., $BERT_{MEDIUM}$, and is pretrained on Legal text from Case Law Project for 1M training steps. Besides the pretraining strategy for adapting BERT to the Legal Domain, we also consider exploiting the tokenization and encoding input by introducing legal terminologies in WordPiece Vocabulary.

Authors of the paper "LEGAL-BERT: The Muppets straight out of Law School" released their Legal BERT Model[6] that is pretrained on legal text from various legal sub-domains. We evaluated their model on legal tasks and compared its performance with our Legal-BERT Models. Comparison is given in the Chapter-6.

---

[6]`https://huggingface.co/nlpaueb/legal-bert-base-uncased`

4 Methods

4.1 Datasets

Legal Datasets used in the pretraining and evaluation process of the Transformer-based Models are taken from the CASELAW Access Project[1] of Harvard Law School. The Legal Data is composed of the Judgments given by the State and Federal Jurisdictions of the Unites-States, on Legal Cases. The CASELAW Access Project's publicly available Legal data is from four Jurisdictions of the states of Arkansas, Illinois, New Mexico, and North Carolina, comprising 358,819 Legal Cases. We acquired further Legal data after applying for access to the researcher's data[2]. Researcher Data includes the Judgements for Legal Cases from all the remaining states and federal Jurisdictions of the United-States. The total Legal Data contains 6,725,065 unique Legal Cases, including Public and Researchers Data. Datasets for Legal Cases are available in `XML` and `JSON` formats. We acquired the JSON format as it fits well with our simple data manipulation and usage requirements. An example of a Legal Case from CASELAW project is given below:

```
1  {
2    "id": 18630,
3    "decision_date": "1997-12-17",
4    "citations": [ { "cite": "124 N.M. 498" } ],
5    "jurisdiction": {
6      "name_long": "New Mexico",
7      "id": 52
8    },
9    "casebody": {
10     "data": {
```

```
11      "judges": [ "APODACA and ARMIJO, JJ., concur." ],
12      "attorneys": [
13        "Maria Garcia Geer, Geer, Wissel & Levy, P.A.,
             Albuquerque, for plaintiff-appellant.",
14        "Michael E. Knight, Knight & Nagel, P.A.,
             Albuquerque, for defendant-appellee."
15      ],
16      "opinions": [
17        {
18          "type": "majority",
19          "text": "OPINION\nPICKARD, Judge. This case
               requires us to consider the appropriate relief
               when a seller wishes to declare forfeiture of a
                large down payment upon default of a real
               estate contract. Caye Buckingham (Buyer) sued
               James Ryan (Seller) for damages resulting from
               .......",
20          "author": "PICKARD, Judge."
21        }
22      ]
23    }
24  }
25 }
```

The Opinion text shown in the example above is truncated to save space; in reality, the Legal Cases' Judgments are much longer, comprising multiple pages. The data for the Legal Cases also includes other information, but the example shows only the relevant fields for sufficient understanding of the Legal Data used.

4.1.1 Pretraining Dataset

The Pretraining process and its benefits are discussed in Section-2.2.3. BERT is pretrained on a large amount of English text from different domains. The details on the pretraining process for BERT are discussed in Section-2.4.2. The pretraining Objectives

used by BERT, i.e., Masked Language Modeling and Next Sentence Prediction, are discussed in Section-2.4.3 and Section-2.4.4, respectively. Pretraining BERT requires plain unlabelled text in a specific format. The input file with unlabelled plain text for pretraining must have one sentence per line, and sentences must be in sequence as they appear in the document. A blank line in the input text file indicates the end of a document and the beginning of a new one.

The Legal Data acquired as described in Section-4.1 contains Opinions from Judges on the Legal Cases from all the state and federal Jurisdictions of the United-States. The text from these Legal Opinions is parsed to generate the pretraining unlabelled data, adherent to BERT's pretraining format requirements. The text from legal opinions is cleaned and pre-processed before splitting it into sentences. NLTK[3] is used for sentence segmentation to create sequential sentences from the much longer legal opinions. To avoid raw, low-quality text, each sentence is then filtered by a set of intuitive rules. Short and low-quality sentences are excluded from the output pretraining data. As per the pretraining text format, a blank line is used as a delimiter to separate multiple documents, as the next sentence prediction objective requires the sentences to follow the actual order in the data. A new document starts when either a Legal Opinion ends, and the next one starts, or one or more sentences get discarded due to the text's low quality. To ensure the quality of documents in the pretraining data, the documents containing fewer than ten sentences are discarded. We provide a simple script[4] for generating pretraining-format adherent data in plain text files, using the Legal Data in `JSON` Format acquired from CASELAW Project[5]. It provides easy pagination on input legal data files and file rotation on output files to generate the pretraining data in an optimal way. The total pretraining data generated from the Legal Cases data from the CASELAW Project constitute about 200 text files, with each file growing on average around 43MB in size.

4.1.2 Legal Opinions Classification Dataset

An example of Legal Case Data from CASELAW Access Project is shown in Section-4.1. It shows opinion from the Judge on the Legal Cases, classified as type `majority`.

---

[3]`https://www.nltk.org/`
[4]`https://gitlab.com/malik.zohaib90/nlp-on-legal-datasets/-/blob/master/utilities/`
`generate_pretraining_data.py`
[5]`https://case.law/bulk/download/`

Similarly, a Judge can dissent from the majority decision of the Court, and in this case, the Opinion is marked as `dissent`. The Opinions in the Legal Cases Data are quite long, often up to multiple pages. However, the limitations of input sequence length for BERT restricts using such long opinions. To solve the sequence length limitation problem, the legal opinions are first summarized using the gensim[6] summarizer. The gensim summarizer is based on the popular TextRank algorithm and is well suited for summarizing long text into important sentences. TextRank is similar to the famous PageRank algorithm, except that it considers text instead of pages.

Opinion	Label
Defendant Eric Galaz appeals from an order revoking his probation based on a violation of a condition of probation prohibiting him from possessing at any time, ...	majority
I concur in every respect with Justice Minzner s dissenting opinion I agree that the State did not meet its burden at the trial level of establishing facts supporting the	dissent

Table 4.1: Legal Opinions Classification Data

Table-4.1 shows an example of the classification data, with truncated legal opinions. On average, the summarization reduces the legal opinions to an average length of around 150 words. The Classification dataset includes about 20K legal opinions, involving approximately 10K for each type of Opinion.

4.1.3 Named-Entity-Recognition Dataset

The Legal Opinions' text from the data acquired from CaseLaw, as described in Section-4.1, is used to prepare Named Entity Recognition Dataset. We used off-the-shelf NLP toolkits to label the Legal data with Named Entities to create silver standard corpora. These corpora were later merged and corrected in a semi-automatic way to a certain extent.

Figure-4.1 shows the complete process for preparing Named-Entity-Recognition data from the Legal Opinions Data. In the first step, text from Legal Opinions is preprocessed

---

[6]`https://radimrehurek.com/gensim/`

Figure 4.1: Preparing NER Dataset

and segmented using the Natural Language Toolkit, i.e., NLTK [LB02]. A plain text file is generated from the Legal Opinions, in a format that is similar to the pretraining data, except the blank line to separate the documents/opinions. In the next step, the Legal Data is annotated in an automated way, using two off-the-shelf NLP toolkits, i.e., NLTK and spaCy[7]. NLTK and spaCy provide a simple and handy implementation for multiple NLP tasks. NER module of NLTK is trained on the Automatic Content Extraction (ACE) corpus [Dod+04] with Maximum Entropy Model [POA16]. NLTK achieves an F1-Score of 89% on the CoNLL NER shared task [SM03]. To label Legal data with Named Entities using spaCy, we use the English multi-task CNN pretrained on OntoNotes [Wei+11], i.e., $en\_core\_web\_sm$[8], with an F1-Score of 85.43%. Once the Legal text is labeled with Named Entities, both Silver standard corpora are merged in an automated way. The automated merging is done using an intuitive mapping between the NER tags of both toolkits, as both toolkits have their own schemes for Named Entity tags, e.g., $ORGANIZATION_{NLTK} = ORG_{SPACY}$, and a simple set of Rules as defined below:

1. Retain the Named Entity (NE) label for a token if both toolkits agree on the same

---

[7]https://spacy.io/
[8]https://spacy.io/models/en#en_core_web_sm

type of Named Entity.

2. Keep the NE label instead of the Non-NE label, i.e., `O` for a token, if only one of the toolkit labels it as a Named Entity.

3. If both toolkits label a token as different Named Entities, keep both, and resolve later during the manual correction.

The cases where toolkits conflicted by labeling a token with different Named Entities were uncommon.

Once the Silver Standard Corpora are merged, we pass it through a recurrent Semi-automatic correction phase. In this phase, first, the disagreements between toolkits are resolved. Afterward, the evident anomalies are observed from the merged and resolved labeled NER data, e.g., *month day, year* not labeled as `DATE`, or Camel-Cased Name preceded by [President, Sir, Dr.] not labeled as `PERSON`. These repetitive patterns are observed and corrected using `regex` in an automated way throughout the Dataset. This process is repeated for all Named Entities types to correct the possible errors in labeling, to get the final Legal Data labeled with Named Entities. The final NER Dataset contains around 59K sentences of Legal text. Figure-4.2 shows a bar chart of the population of Named Entities in the prepared NER Dataset.

4.1.4 Legal Vocabulary

To prepare Vocab-BERT Model, we prepared a Legal Vocabulary defining 1233 legal terms. The legal terms were extracted by crawling the Legal Dictionary[9] using BeautifulSOUP[10]. The Legal Vocabulary[11] prepared is also released along with the Legal Opinions Classification Dataset. The legal-term-frequencies are also added for the legal terms based on the Classification Dataset. For a legal term defined in the vocabulary, legal-term-frequency is defined as the number of Legal Cases from the Classification Dataset, using the legal term. These statistics are necessary to determine the legal term's eligibility for being added in the Vocab-BERT Model, as discussed in more detail in Section-4.2.2.

---

[9]`https://dictionary.law.com/`
[10]`https://www.crummy.com/software/BeautifulSoup/bs4/doc/`
[11]`https://gitlab.com/malik.zohaib90/nlp-on-legal-datasets/-/blob/master/data/legal_`
   `vocabulary/legal_dictionary.tsv`

Figure 4.2: Named Entities Population in NER Dataset

4.2 Transformer-Based Language Models

The research's main focus is the performance comparison of the Transformer Based Language Models in the Legal Domain. Table-4.2 shows the list of employed Models, with their sizes, including the pretrained Models as released by their authors and the ones modified to adapt the Legal Domain.

Model Name	(L, H, A)
BERT-Base-Cased	(12, 768, 12)
XLNet-Base-Cased	(12, 768, 12)
BERT-Medium	(8, 512, 8)
Legal-BERT	(8, 512, 8)
Vocab-BERT	(8, 512, 8)
Legal-Vocab-BERT	(8, 512, 8)

Table 4.2: Transformer Models for Comparison

Where L represents the number of layers, H represents the size of the hidden states, and

A is the number of Attention Heads used in a Model. **BERT-Base-Cased**, as described in Section-2.4, and **XLNet-Base-Cased**, as described in Section-2.5, are comparable in size and configurations. Both Models use the transformer Neural Network as the underlying architecture and use the benefits of pretraining and transfer learning techniques. The major difference between both Transformer-based Models is their pretraining objectives, i.e., BERT being autoencoder and XLNet being autoregressive. BERT-Base-Cased and XLNet-Base-Cased are used as released by authors to evaluate their performance on NLP tasks in Legal Domain, with simple finetuning.

BERT-Medium is a relatively smaller version of the BERT model compared to BERT-Base-Cased. We choose BERT-Medium because it is comparable to the BERT-Base-Cased Model in size and performance, compared to other smaller versions. The main benefit of using BERT-Medium is the ease in the pretraining process as BERT-Base-Cased requires much more memory and pretraining time than BERT-Medium. For this reason, BERT-Medium is for adaptation in the Legal Domain. The three Models prepared during the research, i.e., Legal-BERT, Vocab-BERT, and Legal-Vocab-BERT, are initialized with BERT-Medium Model and inherit the same size and configurations.

Legal-BERT is initialized with BERT-Medium Model and further pretrained on Legal Data, as described in Section-4.2.1. **Vocab-BERT** is BERT-Medium Model fed with Legal Vocabulary, as described in Section-4.2.2. **Legal-Vocab-BERT** is a BERT-Medium Model that combines the powers of both Legal-BERT and Vocab-BERT by adding legal vocabulary to Legal-BERT, as discussed in Section-4.2.3.

Our implementation and experimental setup for performing downstream NLP tasks uses the pytorch implementation of the Transformer Models. **Legal-BERT** is initialized using the tensorflow checkpoint of **BERT-Medium** Model. The output of the pretraining process is the Legal-BERT Model produced as a tensorflow checkpoint. These Models are first converted into pytorch versions using transformers-cli[12] before using on the downstream NLP tasks. Similarly, pytorch versions of **Vocab-BERT** and **Legal-Vocab-BERT** are created to be compared on the NLP tasks on legal data.

---

[12]`https://huggingface.co/transformers/converting_tensorflow_models.html`

4.2.1 Preparing Legal-BERT

BERT-Medium Model is additionally pretrained on 3B tokens of English Legal text from the CASELAW Access Project for 1M training steps to prepare the Legal-BERT Model. The pretraining process for BERT is discussed in Section-2.4.2. BERT is already pretrained on a large amount of unlabelled English text. To adapt BERT to the Legal Domain, we run additional pretraining of the BERT-Medium Model on Legal Data. BERT requires the pretraining data in a particular format. Preparation of Legal Data for pretraining is discussed in Section-4.1.1.

Figure-4.3 shows the preparation of the Legal-BERT Model. The pretraining process involves two major steps, i.e., "Create Pretraining Data" and "Run Pretraining". In the first step, i.e., **Create Pretraining Data**, the unlabelled text, adherent to pretraining format requirements, is converted into TensorFlow-Record (TF-Record) format. This step's output is the TF-Record files, created in a compatible format with the Size and Configuration of the BERT-Medium Model. The second step, i.e., **Run Pretraining**, takes the TF-Record Data as input and pretrains the BERT-Medium Model for 1Million training steps. Both steps, i.e., "Create Pretraining Data" and "Run Pretraining", are performed using bash scripts, facilitated by the python code released by the BERT's authors for running the pretraining process for BERT Model. The output of the second step is the Legal-BERT Model. Legal-BERT was pretrained using GPU "Tesla K80" with 11441MiB Memory. The pretraining process, including "Create Pretraining Data" and "Run Pretraining" steps, took approximately eighteen days on the GPU. Legal-BERT achieved ≈70% and ≈93% accuracies on pretraining objectives, i.e., masked language modeling and next sentence prediction, respectively.

Pretraining BERT is an expensive process, both time-wise and resources-wise. Running out of memory is a commonly faced problem in the pretraining process. Following measures and design decisions are taken to avoid such problems:

1. BERT-Medium Model is used to prepare Legal-BERT as it is relatively smaller, much faster than BERT-Base Model, and requires less memory, yet a good candidate for comparison.

2. Input pretraining text is fragmented into multiple files to avoid loading all the legal data (3B tokens) in the memory at-once.

Figure 4.3: Preparation of Legal-BERT Model

3. During the "Create Pretraining Data" process, files are loaded and produced in a batch of 10 files at a time.

4. The "Run Pretraining" process uses a file glob to read one file at a time from all the TF-Record files.

Parameter	Value
Number of layers (L)	8
Hidden size (H)	512
Attention heads (A)	8
do_lower_case	False
train_batch_size	16
num_train_steps	1000000
num_warmup_steps	20000
learning_rate	2e-5
max_seq_length	128
max_predictions_per_seq	20
masked_lm_prob	0.15
random_seed	12345
dupe_factor	5

Table 4.3: Pretraining Hyperparameters

Table-4.3 shows the Hyperparameters used for pretraining of the Legal-BERT Model. As Legal-BERT is initialized with the pretrained BERT-Medium Model, hence it inherits its size and configurations. **do_lower_case** is set to `False` to make the Legal-BERT Model case sensitive. Maximum sequence length, i.e., **max_seq_length**, is used as 128 tokens because, on average, the sentences are segmented around the same length limit in the pretraining data. **max_predictions_per_seq** shows the maximum number of masked-token predictions BERT should make in a sentence. Masked Language Modeling Probability, i.e., **masked_lm_prob**, is the product of **max_seq_length** and **max_predictions_per_seq** and is set manually, following the BERT's authors' recommendations.

To verify the impact of maximum sequence length, i.e., **max_seq_length** parameter during the pretraining process, we also pretrained BERT-Medium on the same amount of Legal data and hyperparameters, except increasing the maximum sequence length to 256. The pretraining took more than twice the time taken by using the maximum sequence length as 128. Despite the difference in the maximum sequence length, Legal-BERT$_{256}$ did not show any significant improvements over Legal-BERT$_{128}$ on the downstream NLP tasks. We discard the Legal-BERT$_{256}$ from further analysis as it shows similar performance on the downstream NLP tasks and takes more than twice as much time for the pretraining process.

4.2.2 Preparing Vocab-BERT

BERT uses WordPiece Vocabulary to create embeddings, as described in Section-2.4.5. WordPiece Vocabulary included with BERT Model contains around 30K tokens, prepared from the general English text. BERT-Medium Model is fed with additional vocabulary specific to the Legal Domain to prepare Vocab-BERT Model to adapt to the Legal Domain. The preparation of Legal Vocabulary added to Vocab-BERT Model is discussed in Section-4.1.4.

Figure-4.4 shows the preparation of the Vocab-BERT Model. In the first step, Legal Vocabulary is filtered based on its term-frequency in the Legal Data. The term-frequency for a legal term is the number of Legal Opinions using the term in Opinions Classification Dataset. We keep a threshold of 30 Legal Opinions out of the total ≈20K population. To add Legal Vocabulary to BERT-Medium for preparing Vocab-BERT Model, a set of legal terms is created by choosing the legal terms with term-frequency equal to or more than

Figure 4.4: Preparation of Vocab-BERT Model

30. It is important to do such filtering as adding more words in the Vocabulary means the size of token embeddings of the BERT Model increases and makes it computationally more expensive. Intuitively, adding new vocabulary that is not used in our target Data would not contribute towards the improvement in BERT's performance on the target NLP tasks. Once the set of legal vocabulary is prepared, it is added to BERT-Medium Model's Tokenizer. The new vocabulary is added in the Tokenizer without splitting as per the WordPiece tokenization mechanism. A legal term is added in the BERT-Medium Model's WordPiece vocabulary if it is not already included. Eventually, 555 new legal terms are added to the WordPiece vocabulary of Vocab-BERT. In the next step, we update the Parameter, i.e., vocab_size of the BERT-Medium Model. Finally, the vocabulary size and configurations are verified to create the resultant Vocab-BERT Medium Model.

The newly added legal vocabulary in the Vocab-BERT Model affects the tokenization and embeddings generated from the Legal Data. For a legal term that does not already exist in the BERT's WordPiece vocabulary, the BERT tokenizer would greedily break it down until it finds all the tokens in the vocabulary space. However, the same legal-term encountered by Vocab-BERT would be found in its vocabulary and treated as a single token. Following is an example of tokenization and embeddings generated by BERT-Medium and Vocab-BERT, for a legal-term added in Vocab-BERT Model, e.g., "impermissible":

Figure 4.5: Preparation of Legal-Vocab-BERT Model

Input Text: Ethnic discrimination is impermissible by law.

Tokenization:

BERT-Medium: Ethnic   discrimination   is   imp   ##er   ##missible   by   law   .

Vocab-BERT-Medium: Ethnic   discrimination   is   impermissible   by   law   .

Embeddings:

BERT-Medium: 101   27673   9480   1110   24034   1200   27119   1118   1644   119   102

Vocab-BERT-Medium: 101   27673   9480   1110   29622   1118   1644   119   102

4.2.3 Preparing Legal-Vocab-BERT

Legal-Vocab-BERT is created by combining the mechanisms used to create Legal-BERT and Vocab-BERT as described in Section-4.2.1 and Section-4.2.2, respectively. It is equipped with additional pretraining on legal data and additional legal vocabulary. Preparation of Legal-Vocab-BERT is quite similar to the preparation of Vocab-BERT, except that instead of feeding legal vocabulary to BERT-Medium Model, we feed it to Legal-BERT Model prepared by adding pretraining on the Legal Data. After analyzing

the impacts of pretraining BERT on legal data, i.e., Legal-BERT, and adding legal vo-
cabulary to BERT, i.e., Vocab-BERT, we combine both the techniques to analyze the
combined impact on BERT's adaptation to the Legal Domain.

Figure-4.5 shows the preparation of Legal-Vocab-BERT Model. The legal term-frequency
filter, as discussed in Section-4.2.2 for Vocab-BERT, is also applicable to Legal-Vocab-
BERT. We prepare a set of legal terms from the Legal Vocabulary (4.1.4) prevalent in
the Legal Opinions Classification Dataset. These legal terms are added in Legal-BERT
(4.2.1) Tokenizer. Next, the vocab_size of the Legal-BERT Model is updated according
to the number of legal-terms added in vocabulary. Finally, vocabulary size and model
size are verified to produce the Legal-Vocab-BERT Model.

5 Experimentation

5.1 Experimental Setup

Table-4.2 shows six different Transformer [Vas+17] based Models that are evaluated on Natural Language Processing tasks in the Legal Domain. The pretrained Models $XLNet_{BASE}$, $BERT_{BASE}$, and $BERT_{MEDIUM}$, are used as released by their authors. Legal-BERT, Vocab-BERT, and Legal-Vocab-BERT Models are initialized with $BERT_{MEDIUM}$ and adapted to the Legal Domain. The transformers library from huggingface[1] is used to evaluate the Models on NLP tasks. It provides an implementation for most transformer-based models on downstream NLP tasks. $BERT_{BASE}$ and $XLNet_{BASE}$ Models are used from the huggingface repository[2] of the Transformer Models. The remaining Models, i.e., $BERT_{MEDIUM}$, Legal-BERT, Vocab-BERT, and Legal-Vocab-BERT, are first converted from tensorflow checkpoints to pytorch Models before evaluation on the NLP tasks, as we use the pytorch implementation of transformer-based models provided by huggingface. All the models were evaluated on the chosen downstream NLP tasks using Tesla T4 GPU on Google Colaboratory[3].

Experiments are performed multiple times for all the Models on the NLP tasks, on a randomly distributed train, validation, and test Data. The results are taken as an average of ten readings of performance measures for each model and task. The data is split randomly in each assessment to test the model's consistency on any random combination of the input data.

---

[1]https://huggingface.co/transformers/
[2]https://huggingface.co/models
[3]https://colab.research.google.com/

5.2 NLP Tasks in Legal Domain

We evaluate all the models on two different NLP tasks:

1. Sequence Classification Task of Legal Opinions Classification.

2. Token Classification Task of Named-Entity-Recognition on Legal Data.

All the models used in the research are evaluated using the same Datasets and finetuning procedure. Following comparisons between different transformer-based Models are made to answer the research questions defined in Section-1.4:

1. $XLNet_{BASE}$ is compared with $BERT_{BASE}$ to evaluate the bidirectional Transformer Models with different pretraining objectives.

2. $BERT_{MEDIUM}$ is compared with Vocab-BERT to analyze the impact of additional Legal Vocabulary on BERT in the Legal Domain.

3. $BERT_{MEDIUM}$ is compared with Legal-BERT to analyze the impact of additional pretraining on Legal Data on BERT's performance on NLP tasks in the Legal Domain.

4. Legal-Vocab-BERT, Legal-BERT, and Vocab-BERT are compared with pretrained Models $BERT_{BASE}$ and $BERT_{MEDIUM}$ to analyze how well they adapted to the Legal Domain.

5.2.1 Legal Opinions Classification Task

All the Transformer Models described in Section-4.2 are evaluated on the Legal Opinions Classification Task. The Dataset for this task contains the Opinions from Judges on the Legal Cases from the Jurisdiction of the State of New Mexico of the United-States. The Opinions from Judges can be either dissenting or agree with the majority's decision of the Court, hence labeled as `dissent` or `majority`, respectively. Detail on Legal Opinions Dataset is given in Section-4.1.2. The total number of Opinions in the Classification Dataset are 19,927 and almost evenly divided between dissent and majority type Opinions. Table-5.1 shows the population of Opinions in the Classification Dataset.

Figure 5.1: Pipeline for Opinion Classification Task

Opinion Type	Number of Opinions
dissent	9995
majority	9932

Table 5.1: Opinions Population in Classification Dataset

Figure-5.1 shows the pipeline for evaluation of our Models on the Opinions Classification Task. The Opinions Classification Dataset is already preprocessed and summarized to fit the needs of our Models under observation, as described in Section-4.1.2. The classification data is split into Train (70%), Validation (15%), and Test (15%) datasets. Next in the pipeline is selection of the transformer model for finetuning and evaluation. The finetuning process and hyperparameters are kept the same for each model for fair comparison. An Opinion Classifier is fashioned from the Transformer Model's Sequence Classifier by modifying the fully connected output layer for binary classification. The Model under observation is trained and validated for 10 epochs on the training and validation datasets. Visual comparisons between training and validation accuracies for the models used are available on our repository[4]. After training and validation, the best performing model is evaluated on its predictions on the Test data. The details on evaluation measures for all the models are given in Section-6.1.

---

[4]https://gitlab.com/malik.zohaib90/nlp-on-legal-datasets/-/tree/master/nlp/
statistics/classification

Parameter	Value
max_seq_length	256
train_batch_size	16
num_train_steps	8720
num_warmup_steps	0
learning_rate	2e-5
drop_out	0.3
activation_function	gelu

Table 5.2: Finetuning Hyperparameters for Opinions Classification

Table-5.2 shows the finetuning hyperparameters used in the Opinions Classification task for all the models. The maximum sequence length is kept at 256 for the sequence classification task as all the opinions from the Opinions Classification Data are concise to around 150 tokens on average. The batch size is selected as 16. This is the best possible batch size in combination with the maximum sequence length of 256. Given the amount of data, and maximum sequence length, increasing the batch size would result in out-of-memory problem, whereas decreasing it would slow down the training progress significantly. The maximum number of training steps are approximately 8720, based on the size of training data, batch size, and epochs. The dropout probability for the fully connected output layer is set to 0.3. The activation function is used as the default one, i.e., "gelu" for all the models, with a learning rate of 2e-5.

All the Models are trained for 10 epochs to facilitate the analysis of the impact of the pretraining methods and domain adaptation techniques on their computational time. As a stochastic optimization method, for finetuning our models, Adam optimizer, "an algorithm for first-order gradient-based optimization of stochastic objective functions" [KB15] is used. We use the adam optimizer's implementation provided by huggingface[5].

5.2.2 Named-Entity-Recognition Task

The Transformer Models described in Section-4.2 are evaluated on Named-Entity-Recognition (NER) Task. The preparation of NER dataset on Legal data is described in Section-

---

[5]`https://huggingface.co/transformers/main_classes/optimizer_schedules.html#adamw-pytorch`

4.1.3. Table-5.3 shows the population of Named Entity tags in our NER labeled Legal Data. We use the transformers library from Huggingface for performing NER task using our models. It provides both pytorch and tensorflow implementations for Natural Language Processing with Transformer Models. We use the implementation from Stefan Schweter[6] (a forked version of transformers library) to facilitate the Named Entity Recognition task on our Legal Data.

We modified the forked version of transformers library by Stefan Schweter to add classification report for individual Named Entity type. The modified version is available in our repository[7]. Similarly, the transformers library at its version 3.3.0, was tweaked to get around an unhandled exception, which prevented successfully running NER task with XLNet Model. The modified transformers library is available in our resources on the source code repository[8].

Named-Entity (NE) Type	NE Population
PERSON	33293
ORG	27050
DATE	12252
GPE	9753
CARDINAL	1151

Table 5.3: Population of NER Dataset

Figure-5.2 shows the pipeline for running the Named Entity Recognition task on our Legal Data using the Transformer Models from Table-4.2. In the first step, the NER tagged legal data is split into Train (70%), Test (20%), and Dev (10%) datasets. In the next step, the model under observation is chosen from the list of Transformer models involved in the comparison. Finetuning parameters are defined for training the Model. The selected model is used to run the NER task using a utility from the forked version of transformers library from stefan-it[9]. The model is trained for five epochs on the NER Training dataset. Finally, the model is evaluated on the predicted Named Entity labels

---

[6]https://github.com/stefan-it/transformers

[7]https://gitlab.com/malik.zohaib90/nlp-on-legal-datasets/-/tree/master/resources/
transformers_stefan-it

[8]https://gitlab.com/malik.zohaib90/nlp-on-legal-datasets/-/tree/master/resources/
transformers_modified

[9]https://github.com/stefan-it/transformers/blob/master/examples/
token-classification/run_ner.py

Figure 5.2: Pipeline for Named Entity Recognition Task

of the Test Dataset. The details on results of NER task for all the models are given in Section-6.2.

Parameter	Value
max_seq_length	128
train_batch_size	32
num_train_steps	6415
num_warmup_steps	0
learning_rate	5e-5
weight_decay	0
do_train	True
do_eval	True
do_predict	True

Table 5.4: Finetuning Hyperparameters for Named Entity Recognition

Table-5.4 shows the Hyperparameters for finetuning transformer models on the Named-Entity-Recognition task. Since the average length for the sentences tagged for Named Entities in our NER Dataset is less than 128 tokens, the **max_seq_length** is selected as 128. We also verified the impact by performing the NER task using the maximum sequence length as 256, and it didn't show any notable improvements in the evaluation. However, using the maximum sequence length as 256 requires a lot more memory and significantly slows down the models' computation on the NLP task. The **train_batch_size**

is set to 32 for the Named-Entity-Recognition task for all the models. As we have decreased the maximum sequence length to half for the NER task, as compared to Opinion Classification task-5.2.1, we can increase the batch size to double without worrying about the memory usage. The learning-rate is set to 5e-5, as per the recommendations from the Models' authors. The training and validation are run for five epochs for all the models on training and validation NER data. Given the amount of training dataset, five epochs, and batch size, the model is trained for 6,415 update steps. Since the NER token classification utility from the transformer runs the consecutive stages of training, validation, and testing, we keep all three parameters, i.e., do_train, do_eval, and do_predict, to perform training, validation, and testing with it in a single execution.

6 Results

6.1 Legal Opinions Classification Results

The transformer Models used in the research for comparison, from Table-4.2, are evaluated for the Sequence Classification task of Legal Opinions using evaluation measures like accuracy, precision, recall, and f-measure. The population for majority and dissent type of Opinions is fairly even, as shown in Table-5.1.

Model	Accuracy	Training Time
BERT-Base-Cased	94.1471 %	2 hrs
XLNet-Base-Cased	94.2140 %	3 hrs
BERT-Medium	93.9130 %	40 mins
Legal-BERT	95.0836 %	41 mins
Vocab-BERT	94.3478 %	52 mins
Legal-Vocab-BERT	**95.2842** %	51 mins

Table 6.1: Opinions Classification Results

Table-6.1 shows the results for all the Models used for evaluation on the Opinions Classification Task. The accuracy and training time for each Model is average from ten observations with the same finetuning hyperparameters and dataset. The training, validation, and test data are randomly split for each evaluation.

Figure-6.1 shows a comparison of accuracies for different Models on the Opinions Classification Task, using a bar chart. Legal-BERT, Vocab-BERT, and Legal-Vocab-BERT are adapted to the Legal Domain. All the models used in comparison are described in Section-4.2.

Figure 6.1: Legal Opinions Classification Results

	precision	recall	f1-score	support
majority	0.9193	0.9656	0.9419	1452
dissent	0.9659	0.9200	0.9424	1538
accuracy			0.9421	2990
macro avg	0.9426	0.9428	0.9421	2990
weighted avg	0.9433	0.9421	0.9421	2990

Table 6.2: Opinions Classification Report for XLNet-Base-Cased Model

Table-6.2 shows the classification report with different evaluation metrics for the Opinion Classification task using the XLNet-Base-Cased Model.

	precision	recall	f1-score	support
majority	0.9468	0.9346	0.9407	1484
dissent	0.9364	0.9482	0.9423	1506
accuracy			0.9415	2990
macro avg	0.9416	0.9414	0.9415	2990
weighted avg	0.9415	0.9415	0.9415	2990

Table 6.3: Opinions Classification Report for BERT-Base-Cased Model

Table-6.3 shows the classification report with different evaluation metrics for the Opinion Classification task using the BERT-Base-Cased Model.

	precision	recall	f1-score	support
majority	0.9393	0.9380	0.9386	1484
dissent	0.9390	0.9402	0.9396	1506
accuracy			0.9391	2990
macro avg	0.9391	0.9391	0.9391	2990
weighted avg	0.9391	0.9391	0.9391	2990

Table 6.4: Opinions Classification Report for BERT-Medium Model

Table-6.4 shows the classification report with different evaluation metrics for the Opinion Classification task using the BERT-Medium Model.

	precision	recall	f1-score	support
majority	0.9407	0.9616	0.9510	1484
dissent	0.9613	0.9402	0.9507	1506
accuracy			0.9508	2990
macro avg	0.9510	0.9509	0.9508	2990
weighted avg	0.9511	0.9508	0.9508	2990

Table 6.5: Opinions Classification Report for Legal-BERT Model

Table-6.5 shows the classification report with different evaluation metrics for the Opinion Classification task using the Legal-BERT Model.

	precision	recall	f1-score	support
majority	0.9398	0.9468	0.9433	1484
dissent	0.9472	0.9402	0.9437	1506
accuracy			0.9435	2990
macro avg	0.9435	0.9435	0.9435	2990
weighted avg	0.9435	0.9435	0.9435	2990

Table 6.6: Opinions Classification Report for Vocab-BERT Model

Table-6.6 shows the classification report with different evaluation metrics for the Opinion Classification task using the Vocab-BERT Model.

	precision	recall	f1-score	support
majority	0.9450	0.9609	0.9529	1484
dissent	0.9608	0.9449	0.9528	1506
accuracy			0.9528	2990
macro avg	0.9529	0.9529	0.9528	2990
weighted avg	0.9530	0.9528	0.9528	2990

Table 6.7: Opinions Classification Report for Legal-Vocab-BERT Model

Table-6.7 shows the classification report with different evaluation metrics for the Opinion Classification task using the Legal-Vocab-BERT Model.

Our Legal-BERT, Vocab-BERT, and Legal-Vocab-BERT Models have the same size and configurations as BERT-Medium Model. A Legal-BERT Model recently released by Chalkidis et al. [Cha+20], is discussed in the Section-3.7. They pretrained the BERT Model of size and configurations same as BERT-Base, on legal data from different sub-domains of law. We compare their Legal-BERT's performance with our Models adept in the Legal Domain. Figure-6.2 shows a comparison between our family of BERT models specializing in the Legal Domain, with the **Legal-BERT-Base** model from Chalkidis

Figure 6.2: Opinions Classification comparison with Legal-BERT-Base from Chalkidis et al.

et al., on the Opinions Classification Data. BERT-Base-Cased Model in the bar chart is the pretrained general language model.

The detailed results on the classification task, along with the confusion matrices, can be found in our repository[1]. The repository also contains the readings from multiple observations for all the models on the Classification task [2].

6.2 Named-Entity-Recognition Results

We compare all the transformer Models from Table-4.2 on the token classification task of Named-Entity-Recognition. The population of different Named Entities in the NER Dataset is shown in Table-5.3. Since the distribution of Named Entity tags is not even, and the number of Non-Named-Entity tokens is much higher as compared to the Named Entities, making it largely imbalanced, hence we evaluate the Models using f1-score.

---

[1] `https://gitlab.com/malik.zohaib90/nlp-on-legal-datasets/-/blob/master/nlp/results/Classification_Results.md`

[2] `https://gitlab.com/malik.zohaib90/nlp-on-legal-datasets/-/blob/master/nlp/results/best_and_average_scores.txt`

Figure 6.3: Named-Entity-Recognition Results

Model	F1-Score	Training Time
BERT-Base-Cased	**85.6865** %	1 hr 18 mins
XLNet-Base-Cased	85.2921 %	1 hr 45 mins
BERT-Medium	82.8686 %	30 mins
Legal-BERT	82.6935 %	29.5 mins
Vocab-BERT	83.1251 %	2 hrs 5 mins
Legal-Vocab-BERT	82.6127 %	2 hrs 6 mins

Table 6.8: Named-Entity-Recognition Results

Table-6.8 shows the results for all the Models used for evaluation on Named-Entity-Recognition Task on the Test Dataset. The F1-Score and Training Time for each Model is the average of ten observations with the same finetuning hyperparameters and Dataset. The F1-Score is calculated only from the predicted Named-Entity (NE) labels, i.e., excluding predictions of Non-NE tokens.

Figure-6.3 shows a comparison of f1-scores for different Models on Named-Entity-Recognition Task, using a bar chart. Legal-BERT, Vocab-BERT, and Legal-Vocab-BERT are adapted

to the Legal Domain. All the models used in comparison are described in Section-4.2.

	Validation Dataset	Test Dataset
loss	0.0494	0.0596
accuracy	0.9909	0.9897
precision	0.8416	0.8353
recall	0.8416	0.8713
f1_score	0.8534	0.8529

Table 6.9: NER Results for XLNet-Base-Cased Model

Table-6.9 shows the results on Validation and Test Data with different evaluation metrics for the Named-Entity-Recognition task using the XLNet-Base-Cased Model.

	Validation Dataset	Test Dataset
loss	0.0492	0.0595
accuracy	0.9911	0.9898
precision	0.8448	0.8447
recall	0.8619	0.8694
f1_score	0.8532	0.8569

Table 6.10: NER Results for BERT-Base-Cased Model

Table-6.10 shows the results on Validation and Test Data with different evaluation metrics for the Named-Entity-Recognition task using the BERT-Base-Cased Model.

	Validation Dataset	Test Dataset
loss	0.0660	0.0826
accuracy	0.9863	0.9835
precision	0.8290	0.8178
recall	0.8279	0.8399
f1_score	0.8285	0.8287

Table 6.11: NER Results for BERT-Medium Model

Table-6.11 shows the results on Validation and Test Data with different evaluation metrics for the Named-Entity-Recognition task using the BERT-Medium Model.

	Validation Dataset	Test Dataset
loss	0.0657	0.0838
accuracy	0.9861	0.9835
precision	0.8156	0.8176
recall	0.8253	0.8365
f1_score	0.8204	0.8269

Table 6.12: NER Results for Legal-BERT Model

Table-6.12 shows the results on Validation and Test Data with different evaluation metrics for the Named-Entity-Recognition task using the Legal-BERT Model.

	Validation Dataset	Test Dataset
loss	0.0648	0.0786
accuracy	0.9863	0.9838
precision	0.8298	0.8222
recall	0.8286	0.8405
f1_score	0.8292	0.8313

Table 6.13: NER Results for Vocab-BERT Model

Table-6.13 shows the results on Validation and Test Data with different evaluation metrics for the Named-Entity-Recognition task using the Vocab-BERT Model.

	Validation Dataset	Test Dataset
loss	0.0658	0.0822
accuracy	0.9860	0.9838
precision	0.8140	0.8133
recall	0.8218	0.8394
f1_score	0.8179	0.8261

Table 6.14: NER Results for Legal-Vocab-BERT Model

Figure 6.4: NER comparison with Legal-BERT-Base from Chalkidis et al.
.

Table-6.14 shows the results on Validation and Test Data with different evaluation metrics for the Named-Entity-Recognition task using the Legal-Vocab-BERT Model.

Just like the Opinion Classification Task, we also evaluated the Legal-BERT-Base Model released by Chalkidis et al. [Cha+20] on Named-Entity-Recognition Task. Figure-6.4 shows a comparison between our family of BERT models specializing in the Legal Domain, with the **Legal-BERT-Base** model from Chalkidis et al., on the NER Task. BERT-Base-Uncased Model in the bar chart is the pretrained general language model.

The detailed classification report NER task can be found on our repository[3]. The repository also contains the readings from multiple observations for all the models on the NER task[4].

---

[3]https://gitlab.com/malik.zohaib90/nlp-on-legal-datasets/-/blob/master/nlp/results/
NER_Results.md

[4]https://gitlab.com/malik.zohaib90/nlp-on-legal-datasets/-/blob/master/nlp/results/
best_and_average_scores.txt

7 Discussion

Based on the experimentation discussed in Chapter-5 and results shown in Chapter-6, the findings and outcomes are discussed in this Chapter that answers all research questions from Section-1.4. Comparisons between different transformer-based models, as listed in Table-4.2, are discussed. These models include both pretrained models as released by their authors and the models that we customized to adapt to the Legal Domain. All the models are tested on the same datasets, finetuning parameters, and the NLP tasks in the Legal Domain, as discussed in Section-5.2. The results from both NLP tasks, i.e., Opinion Classification and NER, are taken as an average from ten observations by randomly splitting the train, validation, and test data.

7.1 Autoencoder vs Autoregressive Models

Research Question 1: Which of the Transformer based Language Models such as BERT and XLNet perform better in Legal Domain?

The transformer-based pretrained models with different pretraining objectives, i.e., BERT-Base-Cased Model (autoencoder) and XLNet-Base-Cased Model (autoregressive) are compared on the Legal Opinions Classification task and Named-Entity-Recognition task on Legal Data.

Legal Opinions Classification Task: The results from multiple observations show that BERT Model is more stable in performance over the randomly distributed data. XLNet, on the other hand, shows a variation of $\pm 0.8\%$ on randomly split training data. Nonetheless, XLNet Model outperforms BERT Model on the average results taken from multiple observations.

NER Task: The Named-Entity-Recognition task's results, using BERT-Base-Cased and XLNet-Base-Cased Models, show that BERT outperforms XLNet Model on the NER task by a slight margin.

Computational Performance: Given the same amount of training data and size and configurations of the models, XLNet Model takes longer than BERT in performing both Sequence and Token Classification tasks. Besides the computational time, by increasing the input data size, e.g., batch_size, maximum_sequence_length, XLNet Model is likely to run into memory problems before the BERT Model.

7.2 Impact of Legal Vocabulary

Research Question 2: How does Legal Vocabulary affect the performance of BERT on Legal Tech?

Vocab-BERT Model, discussed in Section-4.2.2, is compared with pretrained BERT-Medium Model on the Legal Opinions Classification task and Named-Entity-Recognition task on Legal Data. Vocab-BERT is prepared by feeding the Legal Vocabulary to BERT-Medium Model; hence both models share the same size and configurations. The only difference is that WordPiece Vocabulary of Vocab-BERT includes additional Legal Terms from the Legal Data.

Results from the **Opinion Classification Task** shows that Vocab-BERT consistently shows improvement in performance over the pretrained BERT-Medium Model. Vocab-BERT also performs slightly better than BERT-Medium Model on **Named-Entity-Recognition Task**.

An example of tokenization and embedding using BERT-Medium and Vocab-BERT is given in Section-4.2.2. The additional legal vocabulary, which is prevalent in our target Data, provides the Vocab-BERT with a notable edge over the BERT-Medium Model. The improvement comes from the fact that Vocab-BERT can essentially interpret a longer input sequence than the general BERT model. Vocab-BERT shows a slight improvement on NER task with general Named Entities, as opposed to Legal-BERT. It exhibits the potential for improvements for token classes more specific to Law, e.g., Legal Entities.

Computational Performance: Adding more vocabulary to the existing WordPiece vocabulary increases the vocab_size of the BERT Model. The growing size of vocabulary results in slowing down the Model's training. Vocab-BERT's average training time on Opinion Classification Task showed an increase of approximately 10 minutes when vocabulary is increased by 555 new tokens. The impact on the NER task is even more prominent. Hence it is crucial to add the vocabulary that would contribute to the performance improvements on the target NLP task. It is a necessary trade-off between accuracy and computational performance.

7.3 Benefits of Domain Specific Pretraining

Research Question 3: Is additional pretraining with Legal data on top of an existing pretrained general language model of BERT worthwhile for optimizing NLP performance in Legal Tech?

Legal-BERT Model, discussed in Section-4.2.1, is compared with the pretrained BERT-Medium Model on the Legal Opinions Classification task and Named-Entity-Recognition task on Legal Data. Legal-BERT is prepared by running additional pretraining on Legal Data by initializing it with BERT-Medium Model; hence both models share the same size, configurations, and WordPiece vocabulary.

Opinions Classification Task: Table-6.1 shows the results of all the models used on Opinions Classification Task. Legal-BERT outperforms the BERT-Medium Model by 1.17% on the sequence classification task. Results show that additional pretraining of a smaller BERT Model, i.e., BERT-Medium, on Legal Data, makes it well adapted to Legal Domain than a larger general language model like BERT-Base Model, which is bigger and computationally more expensive.

NER Task: Legal-BERT didn't show any significant improvements on Named-Entity-Recognition Task over the BERT-Medium Model. This behavior is also verified with the Legal-BERT-Base Model, released by Chalkidis et al. [Cha+20], as shown in Figure-6.4. Since NER data tags are generic, additional pretraining on legal data did not gain an advantage on the task. Legal-BERT specializes in the Legal Domain, hence susceptible to better performance on data labeled with legal terms, e.g., Legal Entities, Crimes, Legal Areas, for a token classification task.

Computational Performance: The pretraining process adjusts the model's weights and does not change the size and configuration of the model's architecture. Results show that additional pretraining does not affect the training time of the BERT Model. With the improvements gained from pretraining on legal data, an architecturally smaller BERT model can surpass a larger model that is pretrained on text from the general language on a downstream Legal Task. As a result, our Legal-BERT Model, with the size and configuration of BERT-Medium, outperforms BERT-Base Model and is computationally much faster at the same time, as shown in the Opinions Classification Task's results in the Table-6.1.

7.4 Combining Legal Vocabulary and Pretraining

Vocab-BERT and Legal-BERT showed promising results on the Legal Tasks. It is evident from the Opinions Classification results that additional pretraining on Legal Data yields better performance than additional legal vocabulary. Legal-Vocab-BERT described in Section-4.2.3 is prepared by adding legal vocabulary to Legal-BERT, combining the benefits gained by both techniques. Results from the Legal Opinions Classification task show that Legal-Vocab-BERT is pre-eminent among the family of BERT models adapted to the Legal Domain, as shown in Figure-6.2. Since the Legal-Vocab-BERT model includes the same amount of additional legal vocabulary as Vocab-BERT, its computational time is comparable to Vocab-BERT Model, yet exceeding in evaluation measures.

Competing with Legal-BERT from Chalkidis et al.

Legal-BERT-Base Model is discussed in Section-3.7, which was recently released by Chalkidis et al. [Cha+20]. Authors of the Legal-BERT-Base Model pretrained a BERT model of the size and configurations of the BERT-Base Model on legal text from different legal areas. We compared their Model to our Legal-BERT model, which has the size and configurations of BERT-Medium, a relatively smaller model. When compared on Legal Opinions Classification task, our Legal-BERT Model competes well, with an accuracy of 95.08%, as compared to Legal-BERT-Base with an accuracy of 95.18%. However, our Legal-Vocab-BERT Model, having the size and configurations of BERT-Medium

model, outperforms Legal-BERT-Base Model from Chalkidis et al., on Legal Opinions Classification Task.

The reason behind the better performance of our Legal-BERT and Legal-Vocab-BERT Models compared to the Legal-BERT-Base model from Chalkidis et al., given the size differences, is the idea of "Sub-Domain pretraining". A Model pretrained within a sub-domain of Law outperforms a Model pretrained on text from different sub-domains of Law when performing an NLP task within the given sub-domain. As explained in Section-3.7, the Legal-BERT-Base model is pretrained on legal text from several sub-domains of Law, like legislation, court cases, and contracts. In contrast, our Legal-BERT and Legal-Vocab-BERT Models are pretrained on legal data from the US court cases. The Legal-BERT-Base model prepared by Chalkidis et al. contains 37.3% (4.3 GB out of total 11.5 GB) of the pretraining text from court cases, whereas our Legal-BERT and Legal-Vocab-BERT Models are pretrained on 100% text from US court cases (8.6 GB). The rationale of sub-domain expertise is also proved by Chalkidis et al. in their paper, where multiple variants of the Legal-BERT model specializing in a sub-domain are compared. The authors release multiple variants of the BERT Models specializing in different sub-domains of Law; however, they did not release a model specializing in US court cases.

8 Conclusion

The recent developments in Language Modeling offer a lot of improvements in the field of Natural Language Processing. The invention of the Transformer Neural Network, together with unsupervised pretraining and transfer learning, paved the way for BERT and XLNet Models to achieve state-of-the-art results on various NLP tasks. While these models provide excellent results, further efforts can make them adapt better to a specific domain. We prepared variants of the BERT Model, specializing in the Legal Domain. During the research, we compared these models on multiple NLP tasks in the Legal Domain. We reached the following conclusions based on these models' evaluation, including the pretrained models, as released by their authors and the models specializing in the Legal Domain prepared during the research.

BERT vs XLNet

We compared BERT-Base-Cased and XLNet-Base-Cased Models on Legal Opinions Classification task and NER task on legal data from US court cases. Results show that XLNet outperforms BERT on the sequence classification task of Legal Opinions. However, it is observed that BERT is more stable on the task through multiple experiments. XLNet, on the other hand, deviates by $\pm 0.8\%$ in accuracy over randomly distributed training and test datasets. Results from the token classification task show that BERT performed better on the NER tagged Legal data compared to XLNet Model. Besides the evaluation metrics, BERT is faster than XLNet in the training process, and XLNet is more likely to run into memory issues than BERT when the input data size is increased.

Pretraining BERT on Legal Domain

Legal-BERT showed promising results in the Legal Opinions Classification task, achieving an improvement of 1.17% over the BERT-Medium Model, which is a general language model. The additional pretraining did not affect the computational performance of BERT Model on the downstream NLP tasks. Our Legal-BERT outperformed BERT-Base Model on the Legal Opinions task despite being smaller in size. It shows that Legal-BERT has adapted better in the Legal Domain than its larger, and computationally expensive, general language variant, i.e., BERT-Base. The NER task results depicted that additional pretraining on BERT did not show any improvements on the Task. A token classification task, specific to Law, e.g., Legal Entities Data, will be better suited for the evaluation.

Adding Legal Vocabulary in BERT

Additional Legal Vocabulary helps in achieving improvements on the Legal Tasks. The results from both Legal Opinions Classification and NER tasks showed that Vocab-BERT achieved improvements over the BERT-Medium Model. Vocabulary selection is crucial and must be task-specific, as additional vocabulary slows down BERT's computational performance due to increased vocabulary size. Intuitively, non-existent or less prevalent vocabulary would not positively affect the Model's performance on the target task. Combining the techniques of additional pretraining and legal vocabulary produces better results than models enabled with individual techniques. The Legal-Vocab-BERT model, a combination of Legal-BERT and Vocab-BERT models, produced the best results on the Legal Opinions Classification Task.

Overall, the research questions are answered in the paper by evaluating all the models prepared and used in the research. However, the work opens a path to some additional possibilities for the Model's better adaptation in the Legal Domain. Research shows that additional pretraining on Legal data from US Court Cases makes it better adapted to the Legal Opinions Classification task. A variant of Legal-BERT Model can be prepared by pretraining it on the Legal Data from US Court Cases from scratch. Comparing it with our Legal-BERT Model will answer if the missing learning from other domains positively impacts the Model's performance.

For the preparation of Vocab-BERT Model, additional Legal Vocabulary is included in the existing WordPiece vocabulary of the BERT Model. Although it helps in improving the accuracy, it slows down the computational performance. A variant of Vocab-BERT can be prepared by creating the WordPiece vocabulary from the Legal Data, by keeping the same vocabulary size as the WordPiece vocabulary that comes with pretrained BERT. WordPiece vocabulary can be initialized by the Legal Vocabulary prepared, and then continuing with the remaining vocabulary generated from Legal text to combine both techniques. Comparing it with our Vocab-BERT will show which of the variants perform better on the Legal Tasks.

Legal Opinions Classification's results showed that Legal-BERT outperformed BERT-Base Model, despite being smaller in size and configurations. In the future, with sufficient resources, variants of larger Legal-BERT models can be prepared by pretraining on the Legal Data from US Court Cases. Evaluation of larger Legal-BERT variants will indicate if we can push the accuracy further on the Legal Task.

After the successful results of BERT's domain adaptation, it would be interesting to create similar variants of the XLNet Model. Comparing them with Legal-BERT, Vocab-BERT, and Legal-Vocab-BERT models would show which model have better domain adaptation capabilities.

It is evident from the research that domain adaptation techniques help in improving results within the Legal Domain. It still has lots of possibilities to enhance these models' understanding of the language of Law even better.

A Abbreviations

BERT	Bidirectional Encoder Representations from Transformers
Transformer-XL	Transformer - (Extra Long)
XLNet	Extra Long Network (Named after Transformer-XL)
NLP	Natural Language Processing
RNN	Recurrent Neural Network
LSTM	Long Short Term Memory
CNN	Convolutional Neural Network
GPT	Generative Pre-trained Transformer
ELMo	Embeddings from Language Models
ULMFiT	Universal Language Model Fine-tuning
GPU	Graphics Processing Unit
TPUs	Tensor Processing Units
LM	Language Modeling
MLM	Masked Language Modeling
AE	Autoencoding
AR	Autoregressive
NADE	Neural Autoregressive Distribution Estimation
XML	Extensible Markup Language
JSON	JavaScript Object Notation
NE	Named-Entity
Non-NE	Non Named-Entity
NER	Named-Entity-Recognition
NLTK	Natural Language Toolkit
ACE	Automatic Content Extraction
MiB	Mebibyte
TF-Record	TensorFlow Record
GELU	Gaussian Error Linear Unit

B Code

The Source Code contains implementation for all the datasets and legal vocabulary preparation, running pretraining with BERT, feeding BERT with legal vocabulary, and running Token and Sequence Classification tasks, with all the models used in research. Below is the link to Source Code Repository:

`https://gitlab.com/malik.zohaib90/nlp-on-legal-datasets`

The Models prepared during the research are available at:

`https://gitlab.com/malik.zohaib90/nlp-on-legal-datasets/-/tree/master/models`

C Dataset

The datasets prepared for experimentation on different NLP tasks performed during the research, are available in our repository.

NER Dataset:

https://gitlab.com/malik.zohaib90/nlp-on-legal-datasets/-/tree/master/data/ner

Opinions Classification Dataset:

https://gitlab.com/malik.zohaib90/nlp-on-legal-datasets/-/tree/master/data/classification

The Legal Vocabulary prepared for preparation of Vocab-BERT is available at:

https://gitlab.com/malik.zohaib90/nlp-on-legal-datasets/-/tree/master/data/legal_vocabulary

Bibliography

[BCB14]   Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: (2014). URL: https://arxiv.org/abs/1409.0473 (cit. on p. 10).

[Bas+19]   Valerio Basile et al. "SemEval-2019 Task 5: Multilingual Detection of Hate Speech Against Immigrants and Women in Twitter". In: *Proceedings of the 13th International Workshop on Semantic Evaluation*. Minneapolis, Minnesota, USA: Association for Computational Linguistics, June 2019, pp. 54–63. DOI: 10.18653/v1/S19-2007. URL: https://www.aclweb.org/anthology/S19-2007 (cit. on p. 30).

[BCL19]   Iz Beltagy, Arman Cohan, and Kyle Lo. "SciBERT: Pretrained Contextualized Embeddings for Scientific Text". In: *CoRR* abs/1903.10676 (2019). arXiv: 1903.10676. URL: http://arxiv.org/abs/1903.10676 (cit. on pp. 30, 31).

[Cal+09]   Jamie Callan et al. *Clueweb09 data set*. 2009 (cit. on p. 24).

[Cas+20]   Tommaso Caselli et al. "I Feel Offended, Don't Be Abusive! Implicit/Explicit Messages in Offensive and Abusive Language". English. In: *Proceedings of the 12th Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, May 2020, pp. 6193–6202. ISBN: 979-10-95546-34-4. URL: https://www.aclweb.org/anthology/2020.lrec-1.760 (cit. on p. 30).

[Cas+21]   Tommaso Caselli et al. *HateBERT: Retraining BERT for Abusive Language Detection in English*. 2021. arXiv: 2010.12472 [cs.CL] (cit. on p. 30).

[Cha+20]   Ilias Chalkidis et al. *LEGAL-BERT: The Muppets straight out of Law School*. 2020. arXiv: 2010.02559 [cs.CL] (cit. on pp. 30, 32, 33, 58, 63, 66, 67).

[19]   "Common Crawl". In: (2019). URL: https://commoncrawl.org/ (cit. on p. 24).

Bibliography

[DL15]   Andrew M. Dai and Quoc V. Le. "Semi-supervised Sequence Learning". In: *CoRR* abs/1511.01432 (2015). arXiv: `1511.01432`. URL: `http://arxiv.org/abs/1511.01432` (cit. on p. 9).

[Dai+19]   Zihang Dai et al. "Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context". In: *CoRR* abs/1901.02860 (2019). arXiv: `1901.02860`. URL: `http://arxiv.org/abs/1901.02860` (cit. on pp. 23, 24, 26).

[Dev+18]   Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR* abs/1810.04805 (2018). arXiv: `1810.04805`. URL: `http://arxiv.org/abs/1810.04805` (cit. on pp. 1, 8, 15, 16, 20–22, 26, 29).

[Dod+04]   G. Doddington et al. "The Automatic Content Extraction (ACE) Program - Tasks, Data, and Evaluation". In: *LREC*. 2004 (cit. on p. 38).

[Eth19]   Kawin Ethayarajh. "How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings". In: *CoRR* abs/1909.00512 (2019). arXiv: `1909.00512`. URL: `http://arxiv.org/abs/1909.00512` (cit. on p. 9).

[Gra13]   Alex Graves. "Generating Sequences With Recurrent Neural Networks". In: *CoRR* abs/1308.0850 (2013). arXiv: `1308.0850`. URL: `http://arxiv.org/abs/1308.0850` (cit. on p. 7).

[He+15]   Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015). arXiv: `1512.03385`. URL: `http://arxiv.org/abs/1512.03385` (cit. on p. 12).

[HR18]   Jeremy Howard and Sebastian Ruder. "Fine-tuned Language Models for Text Classification". In: *CoRR* abs/1801.06146 (2018). arXiv: `1801.06146`. URL: `http://arxiv.org/abs/1801.06146` (cit. on p. 8).

[HKC19]   Jerrold Soh Tsin Howe, Lim How Khang, and Ian Ernst Chai. "Legal Area Classification: A Comparative Study of Text Classifiers on Singapore Supreme Court Judgments". In: *CoRR* abs/1904.06470 (2019). arXiv: `1904.06470`. URL: `http://arxiv.org/abs/1904.06470` (cit. on p. 28).

[Hus+20]   Omar Hussein et al. "NLP_Passau at SemEval-2020 Task 12: Multilingual Neural Network for Offensive Language Detection in English, Danish and Turkish". In: July 2020 (cit. on p. 28).

[KB15]   Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Op-
           timization". In: *3rd International Conference on Learning Representations,
           ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Pro-
           ceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://
           arxiv.org/abs/1412.6980 (cit. on p. 51).

[KR18]   Taku Kudo and John Richardson. "SentencePiece: A simple and language
           independent subword tokenizer and detokenizer for Neural Text Processing".
           In: *CoRR* abs/1808.06226 (2018). arXiv: 1808.06226. URL: http://arxiv.
           org/abs/1808.06226 (cit. on p. 24).

[Lee+19]   Jinhyuk Lee et al. "BioBERT: a pre-trained biomedical language represen-
           tation model for biomedical text mining". In: *CoRR* abs/1901.08746 (2019).
           arXiv: 1901.08746. URL: http://arxiv.org/abs/1901.08746 (cit. on
           pp. 29, 30).

[LB02]   Edward Loper and Steven Bird. "NLTK: The Natural Language Toolkit".
           In: *Proceedings of the ACL-02 Workshop on Effective Tools and Method-
           ologies for Teaching Natural Language Processing and Computational Lin-
           guistics*. Philadelphia, Pennsylvania, USA: Association for Computational
           Linguistics, July 2002, pp. 63–70. DOI: 10.3115/1118108.1118117. URL:
           https://www.aclweb.org/anthology/W02-0109 (cit. on p. 38).

[MBG19]   Jelena Mitrović, Bastian Birkeneder, and Michael Granitzer. "nlpUP at
           SemEval-2019 Task 6: A Deep Neural Language Model for Offensive Lan-
           guage Detection". In: Jan. 2019, pp. 722–726. DOI: 10.18653/v1/S19-2127
           (cit. on p. 28).

[Par+11]   Robert Parker et al. "English Gigaword Fifth Edition". In: (2011) (cit. on
           p. 24).

[Pet+18]   Matthew E. Peters et al. "Deep contextualized word representations". In:
           CoRR abs/1802.05365 (2018). arXiv: 1802.05365. URL: http://arxiv.
           org/abs/1802.05365 (cit. on pp. 8, 15).

[POA16]   Alexandre Pinto, Hugo Gonçalo Oliveira, and Ana Oliveira Alves. "Com-
           paring the Performance of Different NLP Toolkits in Formal and Social Me-
           dia Text". In: *5th Symposium on Languages, Applications and Technologies
           (SLATE'16)*. Ed. by Marjan Mernik, José Paulo Leal, and Hugo Gonçalo
           Oliveira. Vol. 51. OpenAccess Series in Informatics (OASIcs). Dagstuhl, Ger-
           many: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016, 3:1–3:16.

ISBN: 978-3-95977-006-4. DOI: `10.4230/OASIcs.SLATE.2016.3`. URL: `http://drops.dagstuhl.de/opus/volltexte/2016/6008` (cit. on p. 38).

[Rad+18]   Alec Radford et al. "Improving Language Understanding by Generative Pre-Training". In: (2018). URL: `https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf` (cit. on pp. 8, 9, 15).

[SM03]   Erik F. Tjong Kim Sang and Fien De Meulder. "Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition". In: *CoRR* cs.CL/0306050 (2003). URL: `http://arxiv.org/abs/cs/0306050` (cit. on p. 38).

[Tay53]   Wilson L. Taylor. ""Cloze Procedure": A New Tool for Measuring Readability". In: *Journalism Quarterly* Volume: 30 issue: 4, (1953), pp. 415–433 (cit. on p. 18).

[UMG21]   Stefanie Urchs., Jelena Mitrović., and Michael Granitzer. "Design and Implementation of German Legal Decision Corpora". In: *Proceedings of the 13th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART,* INSTICC. SciTePress, 2021, pp. 515–521. ISBN: 978-989-758-484-8. DOI: `10.5220/0010187305150521` (cit. on p. 29).

[Uri+16]   Benigno Uria et al. "Neural Autoregressive Distribution Estimation". In: *Journal of Machine Learning Research* 17.205 (2016), pp. 1–37. URL: `http://jmlr.org/papers/v17/16-272.html` (cit. on p. 25).

[Vas+17]   Ashish Vaswani et al. *Attention Is All You Need.* 2017. arXiv: `1706.03762` `[cs.CL]` (cit. on pp. 2, 7–11, 13–15, 22, 23, 48).

[Wei+11]   Ralph Weischedel et al. "OntoNotes: A Large Training Corpus for Enhanced Processing". In: Jan. 2011 (cit. on p. 38).

[Wu+16]   Yonghui Wu et al. *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation.* 2016. arXiv: `1609.08144` `[cs.CL]` (cit. on pp. 2, 19).

[Yan+19]   Zhilin Yang et al. "XLNet: Generalized Autoregressive Pretraining for Language Understanding". In: *CoRR* abs/1906.08237 (2019). arXiv: `1906.08237`. URL: `http://arxiv.org/abs/1906.08237` (cit. on pp. 1, 9, 22, 25, 26, 29).

Bibliography

[Zam+19]   Marcos Zampieri et al. "SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval)". In: *CoRR* abs/1903.08983 (2019). arXiv: 1903.08983. URL: http://arxiv.org/abs/1903.08983 (cit. on p. 30).

[Zhu+15]   Yukun Zhu et al. "Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books". In: *CoRR* abs/1506.06724 (2015). arXiv: 1506.06724. URL: http://arxiv.org/abs/1506.06724 (cit. on pp. 15, 24).