Project No: A3066-221

Knowledge-Enhanced Natural Language Processing in Patent Retrieval

Submitted by: Tao Chen (U1920805B)
Supervisor: Professor Chen Li Hui

School of Electrical and Electronic Engineering

A final year project report presented to the Nanyang Technological University

in partial fulfillment of the requirements of the degree of

Bachelor of Engineering

2023

Table of Contents

Abstract **i**
Acknowledgements . . . ii
Acronyms . . . iii
List of Tables . . . iv
List of Figures . . . vi

1 Introduction **1**
1.1 Motivations . . . 1
1.2 Scope . . . 2
1.3 Organization . . . 3

2 Literature Review **4**
2.1 Natural language inference . . . 4
2.1.1 Models and approaches . . . 4
2.1.2 Benchmark datasets . . . 6
2.2 Knowledge integration . . . 8
2.2.1 Input-focused injections . . . 8
2.2.2 Architecture-focused injections . . . 9
2.3 Knowledge distillation . . . 10
2.4 Information retrieval . . . 12

3 Preliminary Study on Patent Pair Modeling **14**
3.1 Patent datasets . . . 14
3.1.1 Patent similarity search dataset . . . 15
3.1.2 Patent match dataset . . . 15
3.2 Method 1: Knowledge Infusion . . . 16
3.2.1 Methodology . . . 16
3.2.2 Ablation Study . . . 20
3.3 Method 2: Joint Optimization of Multiple Loss Functions . . . 22
3.3.1 Methodology . . . 22
3.3.2 Experimental results and analysis . . . 23
3.4 Method 3: Self-Distillation of Encoders . . . 26
3.4.1 Methodology . . . 26
3.4.2 Experimental results and analysis . . . 28

4 K-PAR: The Knowledgeable Prior Art Retriever **30**
4.1 Methodology . . . 30
4.2 Experimental setup . . . 31
4.2.1 Test set formulation . . . 31
4.2.2 Evaluation metrics . . . 31
4.2.3 Indexing and quering . . . 32

4.3 Experimental results and analysis 32

5 Future Work **36**
5.1 Open-domain question answering 36
5.2 Knowledgeable adapter pre-training 36
5.3 Memory-efficient storage 37
5.4 Human-in-the-loop for continual improvement 37

6 Conclusion **38**
Reflection on Learning Outcome Attainment 39
Appendix 39

A Software Environment and Required Packages **40**

B Loss functions and pooling methods for joint optimization **41**
B.0.1 Binary cross entropy 41
B.0.2 Mean squared error 42
B.0.3 InfoNCE 42
B.0.4 CLS and MEAN pooling 43
Bibliography 43

Abstract

Document retrieval has long been a complex research problem. In the context of patents, we use document retrieval to search all novelty-destroying prior art given a patent application: a database of the prior art is assessed against a new patent to determine if the patent office can grant an exclusive right to an invention, product, or process. Therefore, patent document retrieval is essentially a matter of semantic comparison or Natural Language Inference (NLI).

This project aims to solve a problem of this kind, namely patent similarity search, whose task is to identify the degree of correspondence between a patent application and prior art. The objective of the proposed project is to outperform the current best on the patent similarity search dataset by incorporating external knowledge while retaining the power of the pre-trained language model.

In this paper, we proposed a structure that incorporates domain-specific and linguistic information into plug-and-go adapter modules attached to a bi-encoder. We first introduced past research work and Natural Language Processing (NLP) models in the literature review. Then, we construct the baseline and perform empirical studies to analyze the consequences of different model designs. We show the effectiveness of our model architecture and extend our model from patent-pair similarity comparison to prior art retrieval. Finally, we suggest areas for future improvement and conclude our findings.

Acknowledgements

I want to express my sincerest gratitude to the following people for their support throughout this FYP project.

Firstly, I would like to thank Professor Chen Lihui for providing me with constructive advice and valuable guidance during the ideation stage of this project. Moreover, I highly appreciate her arranging biweekly meetings with me in her busy schedule to discuss my progress and research directions. Her supervision is essential to this project right from the start of research scope formulation to implementation.

My next thanks go to Dr. Charlie Soh Zhan Yi for offering me relevant materials and resources. His suggestions provide invaluable input to my project, and he also helps me a great deal with model technicality. Last but not least, I want to thank the Machine Learning and Data Analytics lab at NTU for granting me GPU usage to conduct experiments.

Acronyms

AUC Area under the ROC Curve.

BCE Binary Cross Entropy.

KG Knowledge Graph.

LSTM Long Short-Term Memory.

mAP Mean Average Precision.

MLM Masked Language Modeling.

MRR Mean Reciprocal Rank.

MSE Mean Squared Error.

MTL Multi-task Learning.

NLI Natural Language Inference.

NLP Natural Language Processing.

NLU Natural Language Understanding.

PLM Pre-trained Language Model.

List of Tables

2.1 **NLI benchmark datasets** 6

3.1 **Baseline Hyperparameters** 20

3.2 **Baseline performance and preliminary experiements on Patent-Match** 20

3.3 **Baseline performance and preliminary experiements on Patent-Match** 21

4.1 **Retrieval evaluation results of bi-encoders** initialized with different (pre-trained model, adapters) checkpoint pairs saved in our implementations in chapters 3 to 3.4 33

4.2 **Retrieval evaluation results of cross-encoders** initialized with different (pre-trained model, adapters) checkpoint pairs saved in our implementations in chapters 3 to 3.4 35

4.3 **A full-text example of prior art retrieval** 35

List of Figures

2.1 **Structural comparison between bi-encoders and cross-encoders** 5

2.2 **An illustration of the teacher-student network used in knowledge distillation.** 11

2.3 **Query-document matching diagram in neural information retrieval.** 12

3.1 **A visual conceptualization of baseline model.** TRM represents the transformer layer while KIA represents the adapter layer 17

3.2 **Structure of the adapter layer.** The number of transformer layers in each adapter layer defaults to two. 17

3.3 **The bi-encoder model architecture for joint optimization.** The final loss is the sum of the factual loss and the prediction loss. 23

3.4 **Training loss by different combinations of loss functions and pooling methods on a bi-encoder over 8 epochs** 24

3.5 **Evaluation AUC by different combinations of loss functions and pooling methods on a bi-encoder over 8 epochs** 24

3.6 **Evaluation AUC by different loss functions under CLS pooling on a bi-encoder.** BCE is applied to 20 epochs while InfoNCE and MSE are applied to 12 epochs as they cannot improve further beyond the 12th epoch. 25

3.7 **Evaluation accuracy with CLS pooling and BCE loss on a bi-encoder on PatentMatch over 20 epochs** 26

3.8 **An illustration of our self-distillation method.** 27

3.9 **Bi-encoder Distillation Performance Comparison.** Except the initial bi-encoder is trained over 20 epochs, the other two self-distilled bi-encoders are trained over 2 epochs. The training trend of the initial bi-encoder is only shown till the 12th epoch for better rendering. . . . 28

3.10 **Cross-encoder Distillation Performance Comparison.** Except the initial bi-encoder is trained over 4 epochs, the other two self-distilled cross-encoders are trained over 3 epochs. The training trend of the vanilla bi-encoder is only shown till the 3th epoch for better rendering. 28

Chapter 1

Introduction

1.1 Motivations

We have witnessed significant advances in the realm of NLP since 2017 when the transformer [1] architecture was introduced. Variants of the original model setup and novel implementations spring up and usher in a new era of commercialized NLP tools. The models developed during this period are broadly classified as auto-encoding and auto-regressive, with BERT [2] being a paragon of the first kind and GPT-3 [3] of the latter.

While the impacts of these powerful models are evident to all, they struggle to escape the tragedic effect of catastrophic forgetting, meaning that the newly learned information drastically washes off the previously learned one. Such a problem makes language models especially hard to retain factual knowledge and common sense. It is especially true for patent data, where domain knowledge hidden in patent texts differs from the semantic relationships learned at the pre-training stage. Therefore, naively fine-tuning pre-trained models on patent texts would not give promising results. Language models must incorporate domain knowledge while retaining past linguistic wisdom to solve this problem. Hence, our study aims to design a novel method that facilitates knowledge infusion and trains a robust model that automates the inference of patent text similarity.

The purpose of this work is to study knowledge integration on domain-specific

tasks. We specifically choose patent data as our dataset of interest because it contains legal jargon and specialized terminology, which intuitively requires external knowledge to analyze.

The bottleneck for the current NLP research toward patent documents has always been the lack of reference to specialized terms. Despite the need to inject additional knowledge, most prior works directly fine-tune Pre-trained Language Model (PLM) without linking entities mentioned in the patent documents to external resources. Unlike those, this project infuses knowledge in adapter modules, which are more lightweight to train and plugged outside the PLM. We build our work on K-adapter [4], a novel model architecture that incorporates knowledge in adapters in an easy plug-and-go way. This project will investigate mechanisms that more effectively integrate language with external knowledge and how to build a knowledgeable retrieval system for patent data.

1.2 Scope

In light of the complexity of manual examination of all prior arts, an effective computerized patent retrieval system is of noteworthy help to the patent office. Our main objective is to investigate and compare different knowledge integration techniques and design a novel model structure catering to patent text and generalizable to other natural language tasks requiring domain knowledge.

Firstly, the patent similarity search dataset is filtered and preprocessed to become tougher. Then, we construct our baseline by training knowledgeable adapters and attaching them to the PLM for fine-tuning. Ablation studies and the effects of different model settings are tested and analyzed empirically. Moving on from the baseline, we introduce additional training objectives to ground the language model to domain knowledge, and further enhance the capability of the model by iteratively training the bi-encoder and its cross-encoder counterpart in a self-distillation way. We also propose a prior art retriever K-PAR that retrieves the most related prior art for new patent applications.

1.3 Organization

The flow of this project is organized as follows. We start with a literature review and introduce proven models and techniques in chapter 2. Next, we describe the datasets of use, analyze baseline results, and conduct various experiments to study the effects of model setups in chapter 3. We present the methodology, then show experimental results and discuss the implications. Section 3.3 discusses the adaptation of our baseline into a bi-encoder and the learning outcome of a newly-designed training objective. We examine a training strategy similar to the teacher-student knowledge-distillation method in section 3.4. Next in chapter 4, we talk about our prior art retriever K-PAR and its performance analysis. Finally, we discuss potential improvements for future work in chapter 5 and conclude our work in chapter 6.

Chapter 2

Literature Review

2.1 Natural language inference

Natural-language inference (NLI), or Natural Language Understanding (NLU), is a subtopic of natural language processing. It is the task of determining whether the given "hypothesis" logically follows from the "premise." Given a pair of hypothesis and premise, their possible relationship may be one of entailment, contradiction, or neutrality. Though the possibility of relationships is not limited to the above-mentioned three for some tasks, and different tasks can have slight modifications in the definition of entailment/contradiction, the basic idea is similar.

2.1.1 Models and approaches

In the post-transformer phase, the typical method to approach such a problem is to pass the joint representation of both sentences to be compared into an auto-encoder model, such as BERT or DeBERTa [5]. These two sentences shall be structured as follows

$$< CLS > sentence1 < SEP > sentence2$$

where <CLS> denotes the classification head and <SEP> is the separation token. The inference result is taken from the output of the <CLS> token, generally passed down to another linear layer and a softmax layer for decision.

It is also possible to train two identical models using a bi-encoder (aka. a Siamese

[6] structure). Bi-encoder is a class of neural network that contains two or more identical sub-networks and all of them share the same weights during training. Two sentences are passed separately to one model and their similarity levels are measured at the output using cosine similarity score. However, bi-encoders usually under-perform cross-encoders on sentence comparison tasks. This is because cross-encoders can exploit the attention mechanism to derive inter-sentence relations for better performance. However, any two sentences require one pass to the model, so cross-encoders are computationally more expensive. On the other hand, bi-encoders find semantic relationships between sentences only through a similarity measure, but they are more convenient as they only need a single pass for each sentence to produce an embedding, which can then be stored in a database and compared with others. A diagrammatic comparison between bi-encoders and cross-encoders can be found in Fig 2.1

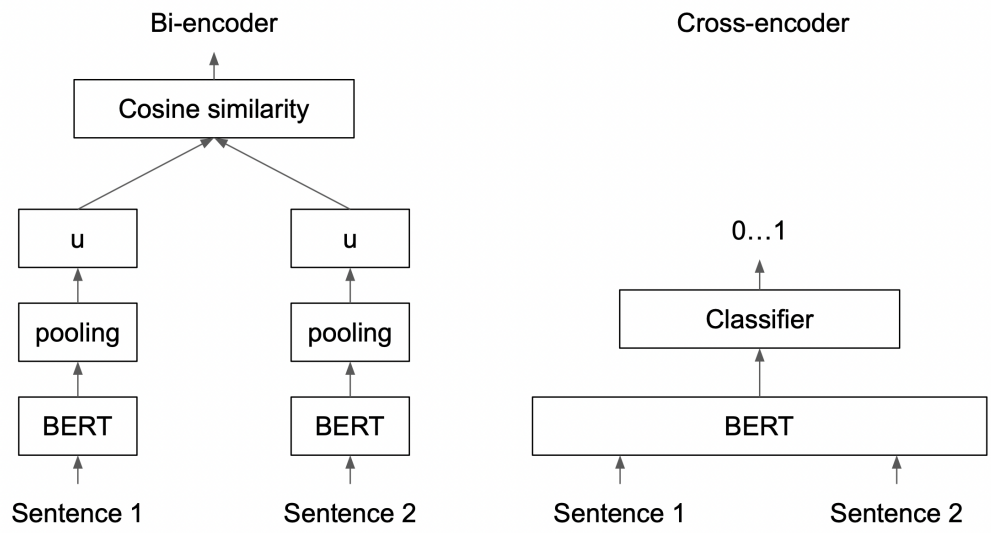

Figure 2.1: **Structural comparison between bi-encoders and cross-encoders**

Other techniques like Multi-task Learning (MTL) [7] and prompting are also on the rise. MTL builds on the intuition that what is learned from one task can be applied to another and learns tasks in parallel in a shared representation, resulting in a regularization effect that improves the generalizability of the model. MT-DNN [8] adopts MTL by optimizing with a joint loss across multiple NLU tasks. It performs

well on ten NLU tasks across popular benchmarks and is proven effective in domain adaptation experiments.

Radford *et al.* [9] shows that language models are unsupervised learners if presented with naturally occurring prompts. This characteristic is essential to zero-shot and few-shot learning, where no or a few prompts (i.e. demonstrations) are used in place of a whole lot of training data. PET [10] leverages empirically-tested discrete prompts to fit NLI sentence pairs into a fixed pattern and a verbalizer for prediction. They show that besides the original unsupervised setting, providing task descriptions to PLMs can aid regular supervised learning, specifically for applications in NLI.

2.1.2 Benchmark datasets

Given the many benchmarks and datasets in NLI, we systematically list and compare the most popular ones in the Table 2.1.

Table 2.1: **NLI benchmark datasets**

Year	Dataset	Corpus composition	Labels
2014	SICK [11]	Sentence pairs generated from existing sets on image captions and video descriptions	Entailment, contradiction, and neutral
2015	SNLI [12]	Premises are based on image captions and hypotheses are created manually according to different labels.	Entailment, contradiction, and neutral
2017	SciTail [13]	Extracted from multiple-choice science exams and web sentences	Entailment, neutral
2018	MultiNLI [14]	Modeled after the SNLI dataset schema but offers texts from ten different genres.	Entailment, contradiction, and neutral

Continued on next page

Table 2.1 – NLI benchmark datasets

Year	Dataset	Corpus composition	Labels
2018	FEVER [15]	Each example consists of a URL of the Wikipedia page containing the premise and a paraphrased or mutated facts from that Wiki page as the hypothesis	Supported, refuted, not enough info
2018	QNLI	Part of GLUE [16] and consists of question-answer pairs altered from the question-answering dataset SQuAD [17]	Entailment, not entailment
2018	WNLI	Part of GLUE [16] and altered from the Winograd Schema reading comprehension challenge by substituting ambiguous pronouns with possible referents	Entailment, not entailment
2019	RTE	Part of GLUE/SuperGLUE [18] and collected from competitions on textual entailment.	Entailment, not entailment
2019	CB	Part of SuperGLUE [18] and drawn from Wall Street Journal, British National Corpus, and Switchboard.	Entailment, not entailment, unknown
2020	ANLI [19]	Generated via Human-And-Model-in-the Loop Entailment Training (HAMLET), hence more difficult than others by design.	Entailment, contradiction, and neutral

2.2 Knowledge integration

Knowledge Graph (KG) is a structured graph comprised of nodes and edges, with each node being an entity and each edge being an ontology or relation. We can represent a KG as a collection of triples (S, R, O), where S is the subject, R is the relation, and O is the object. An example assertion is “papaya is a fruit" and its representation as a tuple would be: (papaya, is_a, fruit). Broadly speaking, a knowledge graph records a network of entities, their properties, semantic types, and relationships. We would like to "inject" this structured knowledge into contextual language models, and this injection would enable the model to integrate information from the KG to improve its performance at inference time. Different approaches have been taken to achieve knowledge injection. Though termed "knowledge graph," KG does not necessarily come in a graph form. KG can be processed by a graph neural network if represented by a graph, or it can also be the direct input into a language model when restructured into sentences. Knowledgeable language models incorporate this knowledge graph concept and thus have more powerful inference ability on domain-specific data.

According to the classification introduced by Colon-Hernandez et al. [20], knowledgeable pre-trained language models are broadly grouped into input-focused injections and architecture-focused injections. We thus discuss previous work in this section according to these two categories. Combinations among them will also be explored.

2.2.1 Input-focused injections

Input-focused injections integrate knowledge by augmenting input as external features or extending language models to accept knowledge graph input. A representative work is ERNIE (Enhanced Language Representation with Informative Entities.) [21] Considering outside knowledge can benefit the understanding of low-resource entities, ERNIE encodes entities into embeddings as an additional input alongside the original word embeddings. This structure learns text dependencies at lower lay-

ers and gradually integrates knowledge about the entities at upper layers. In the pre-training stage, link prediction and Masked Language Modeling (MLM) are the joint objectives for knowledge integration. ERNIE improved performance on entity typing and relation extraction datasets in entity-rich settings and demonstrated the efficacy of incorporating knowledge into language models.

K-BERT [22] directly adds attribute entities as triples to the input of the PLM. Positional embeddings are added to the injected triples and fed to the model in between the actual text inputs. The resulting text input forms a hierarchical tree structure in which a specifically designed attention mask applies to distinguish relations between entities. The role of such a attention mechanism is to mimic the graph connections.

KELM (Knowledge-Enhanced Language Model) [23] is another example of knowledge augmentation. It verbalizes the entire English Wiki-data by converting the KG triples into natural texts. The KLEM allows an easier knowledge integration for existing language models as KG information is in text form, the same modality as the natural language corpora.

2.2.2 Architecture-focused injections

Architecture-focused injections establish additional knowledge-guided pre-training tasks or neural architecture to regularize the learning objective. Inspired by Xie *et al.* [24], KEPLER [25] uses entity descriptions to construct entity embeddings. They encode the texts and entities into a unified PLM as the encoder and optimize the combined KE and MLM objectives.

Unlike all models mentioned above, KGLM [26] utilizes Long Short-Term Memory (LSTM) [27] instead of a transformer-based model. Its key idea is to condition a language model on a knowledge graph by predicting the next word or entity relevant to the sentence. KGLM builds a local knowledge graph as it iterates through the sentence, which is just a subset of the full KG.

KnowBert [28] pre-trains an integrated entity linker (EL) as an extension to BERT through a mechanism called Knowledge Attention and Re-contextualization

component (KAR). KnowBert tries to figure out the mentions or candidates of entities, so entity annotations are not required. It shows better performance over ERNIE on downstream tasks due to the introduction of the multi-task training regime.

Retrieval-based knowledgeable PLM has also gained research attention. REALM [29] augments language model pre-training with a neural knowledge retriever that retrieves knowledge from an external corpus. Given an input and a retrieved document, the knowledge-augmented encoder then performs cross-attention to determine the entity at the masked position. To address the computational challenge of document retrieval, REAL employs asynchronous MIPS refresh to find the approximate top-k documents.

Adapters [30] seem to be a promising research field to incorporate knowledge as well. Due to its compact and extensible features, the adapter can combine various kinds of knowledge into one system. An example is the K-adapter [4], a modification of RoBERTa with adapter modules to infuse knowledge. K-adapter is made up of adapter plug-ins and backboned by RoBERTa-large. During pre-training, the adapter modules are trainable while RoBERTa remains frozen. On the contrary, the RoBERTa is trainable, while adapters are frozen during fine-tuning. In contrast to previous models, K-adapter does not cause catastrophic forgetting of previously learned information and even achieves better results when new kinds of knowledge are introduced. Our work is based on K-adapter, and therefore falls under the category of architectural injection.

2.3 Knowledge distillation

There is a growing interest in academia to obtain fast and lightweight models that retain similar performance to large-scale models like GPT-3 [3], which come with a staggering cost of training. It often can be achieved by knowledge distillation, a technique to transfer knowledge from a more complex teacher network to a simpler student network. One of its applications involves compressing complex models into

more compact ones suitable to run on constrained devices such as smartphones or embedded systems. There are several ways to perform knowledge distillation, but the basic idea all boils down to training the student model that mimics the outputs of the teacher model.

Hinton, Dean, and Vinyals [31] proposed using the teacher-student framework to distill knowledge in a neural network in 2015. In training the student model, the student objective function is a weighted average of two losses, one with the hard targets (the ground truth labels) and the other with the soft targets (in probabilities) produced by the teacher model. Adding these soft targets instructs the student model to learn from the class distribution containing richer information than just the labels. An illustration of their original approach is in figure 2.2.

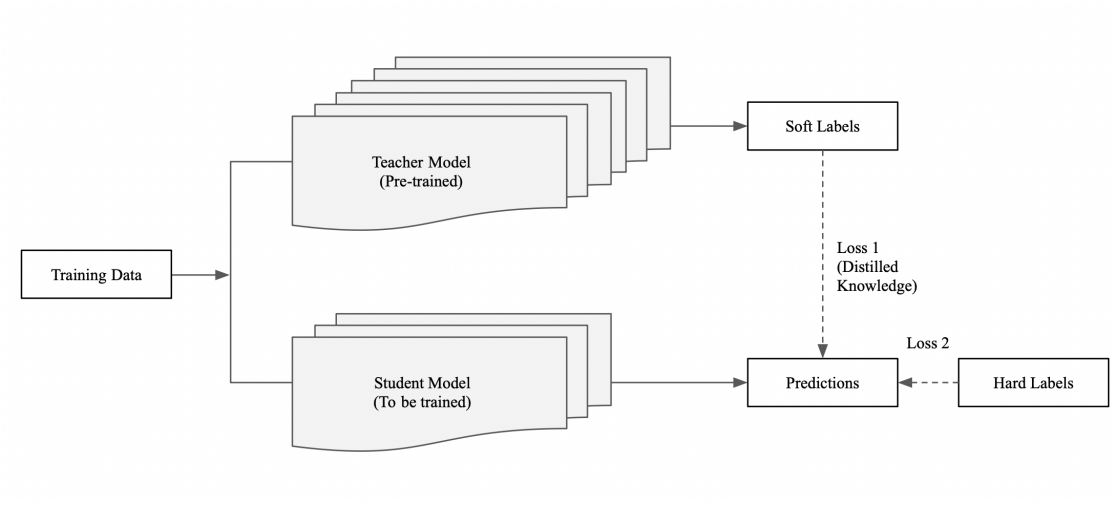

Figure 2.2: **An illustration of the teacher-student network used in knowledge distillation.**

Recent papers have explored the teacher-student technique under varying goals other than model compression. For example, STraTA [32] reinforces the few-shot learning ability of BERT by iteratively fine-tuning it on an auxiliary task before the target task. In each round, the student model learns from the pseudo-labels generated from the teacher model. The learning process repeats until it meets some stopping criteria.

Trans-encoder [33] deploys similar strategies to train cross- and bi-encoders alternately until convergence. They start from a reasonably robust bi-encoder model by transforming a PLM using contrastive learning. Then they perform self-distillation

for several cycles, each training a cross-encoder with bi-encoder-labeled data and vice versa. ERNIE-search [34] also distills knowledge from a cross-encoder to a bi-encoder. Instead of training both structures alternately, they use what they call the "cascade distillation" to consider different information interaction schemes between the two encoders.

The resulting bi-encoder becomes an effective knowledge retriever. The ways Trans-encoder and ERNIE-search distill knowledge are the most related to our work, so we pass down insights on some design choices from these two papers to our experimentation in chapter 3.4.

2.4 Information retrieval

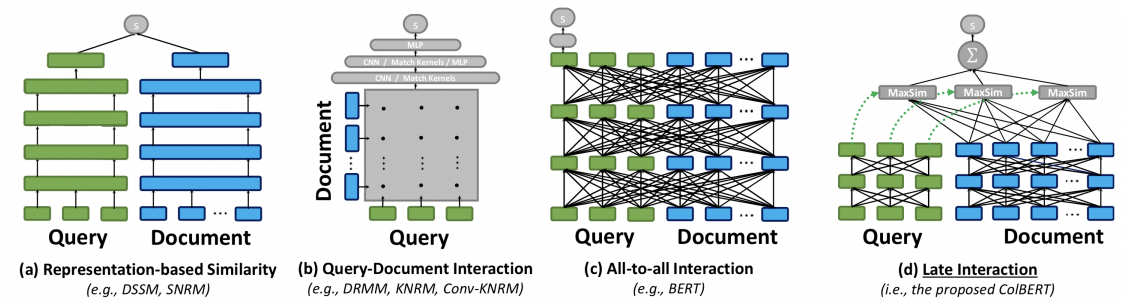

Figure 2.3: **Query-document matching diagram in neural information retrieval.**

Information retrieval is the process of retrieving relevant information from a collection of data or documents based on a user's query. This involves analyzing and indexing documents, constructing a searchable index, and matching the index to user queries to retrieve relevant information. Neural ranking models have been used to extract feature representations of query and document for comparison. There are numerous neural ranking paradigms from past work that exploit a variety of interaction schemes to capture the correlation between two texts. As noted in ColBERT [35], these interaction schemes can be broadly categorized into four types (figure 2.3): representation-based similarity, query-document interaction, all-to-all interaction, and late interaction.

Representation-based similarity directly calculates the relevance between two

embeddings prepared independently for both texts. It is the simplest form of interaction, and its performance largely depends on the quality of the embeddings. Most information retrieval tasks use the representation-based similarity measure for its cheap and fast computation that enables real-time inference.

The query-document interaction models word-level relationships by linking the two input texts by a deep neural network, usually through CNN [36] or MLP (multi-layer perceptron). It is equivalent to learning an interaction matrix of every combination of word pairs from both sentences. The all-to-all interaction advances this concept to include the inter-correlation between words in each sentence. Although this scheme yields the best performance in theory, it also imposes the heaviest computational overheads. The transformer-based cross-encoder follows this design.

According to ColBERT, fine-grained matching of models like transformers can be combined with pre-computation of document representations by judiciously delaying the query–document interaction. They use the MaxSim operators to compute the maximum cosine similarity between each query embedding and all document embeddings and sum these scalar outputs for each query term. This simple yet effective method improves the recall score substantially, as claimed by the original paper. Lu *et al.* [34], similarly, uses cosine similarity to measure the [CLS] output of the two patent texts and a MaxSim on their whole embeddings.

Chapter 3

Preliminary Study on Patent Pair Modeling

3.1 Patent datasets

When searching for prior art for a patent application, patent officers mainly rely on vanilla keyword searches, all of which offer limited semantic context information. Moreover, keyword search does not consider synonyms, and the exact terms must exist in the current database for matching. There have been efforts in constructing patent datasets for the research of patent search automation. In this project, we investigate the following two patent datasets: Patent Similarity Search and Patent Match.

For each dataset, the label indicates the degree of semantic correspondence matching between text pairs. Label 1 implies the two patent documents describe the same thing, or we can say that the cited prior art is novelty-destroying for the Patent Match case. On the contrary, label 0 means the inventions described are dissimilar, or the prior art does not challenge the submitted application for the Patent Match case.

3.1.1 Patent similarity search dataset

The Patent Similarity Search Dataset [37] [1], which we refer to PatentSim as a shorthand, is proposed in 2019 to facilitate patent searching research. The corpus contains patent documents published between 2000 and 2015, with the CPC[2] category A61[3].

The authors provide evaluation results with basic NLP techniques like bag-of-words and doc2vec, in three different settings whereby the full-text, abstracts, or only the claims are used for comparison. PatentSim is much simpler than PatentMatch, for its Area under the ROC Curve (AUC) score reported using the doc2vec method on the abstract section is about 90%. To make the dataset harder, we subset PatentSim to include only 900 pairs for training and 100 pairs for validation, both with balanced labels. Our subsequent mentions of PatentSim refers to this harder version instead of the original dataset, and all evaluation results reported on it are AUC scores.

3.1.2 Patent match dataset

The Patent Match Dataset [38] [4], which we address as PatentMatch for subsequent mentions, follows a different patent classification scheme. Instead of the CPC scheme, patents are classified based on levels of relevance and take on alphabetical names like X, Y, P, A, D, etc. PatentMatch is available in three versions: ultra-balanced, balanced, and regular. We select the ultra-balanced version for all our experiments, which consists of 20,270 pairs of patent texts (90% for training and 10% for validation). The PatentMatch dataset is very challenging and yields much lower results than PatentSim.

[1] The PatentSim dataset is downloadable through this link: `https://figshare.com/articles/corpus_tar_gz/7257194`

[2] CPC: Cooperative Patent Classification, a patent classification system that divides patents into nine sections, A-H and Y, and further segregates into classes and groups, according to their technical fields

[3] Category A61: medical or veterinary science, and hygiene

[4] The PatentMatch dataset is downloadable through this link: `https://hpi.de/naumann/projects/web-science/paar-patent-analysis-and-retrieval/patentmatch.html`

3.2 Method 1: Knowledge Infusion

3.2.1 Methodology

Baseline structure

Inspired by K-adapter [4], we inject multiple kinds of knowledge by training adapters on different pre-training tasks. The factual adapter (facAdapter) was pret-trained on a subset of the relation classification task T-rex [39], while the linguistic adapter (linAdapter) was pre-trained on a dependency relation prediction set from Book Corpus [40]. As the project focus is to investigate the mechanism of effective knowledge fusion and not the adapter pre-training, we directly experiment on the model checkpoints provided by the K-adapter team.

These adapters are external plug-ins that are composed of three layers, each with two transformer sub-layers and two projection sub-layers, and a skip-connection applied across the two projection sub-layers. Specifically, the adapter layers are plugged between different transformer layers of the PLM (the first, eleventh, and twenty-second layers).

The output hidden feature of the transformer layer in the PLM and the output feature of the previous adapter layer are concatenated as the input feature of the current adapter layer. The last hidden features of the PLM and adapter are then concatenated to form the final output feature of the knowledge-specific adapter.

When only one knowledge-specific adapter is used, the final output feature of the adapter model can be taken as the input for the task-specific layers of the downstream task. When multiple knowledge-specific adapters are used, the output features of different adapter modules are concatenated as the input for the task-specific layers of the downstream task.

We use RoBERTa-Large as our backbone PLM and keep it frozen during the adapter pre-training stage, then make it trainable during fine-tuning. The adapter modules are also trainable when we fine-tune the RoBERT-Large. Both patent datasets are encoded using a cross-encoder structure, and the input tokens are struc-

tured as in the next sentence prediction format:

$$< SEP > patent_1 < /SEP > patent_2 < /SEP >$$

See figure 3.1 as the baseline model structure and figure 3.2 for the adapter setup.

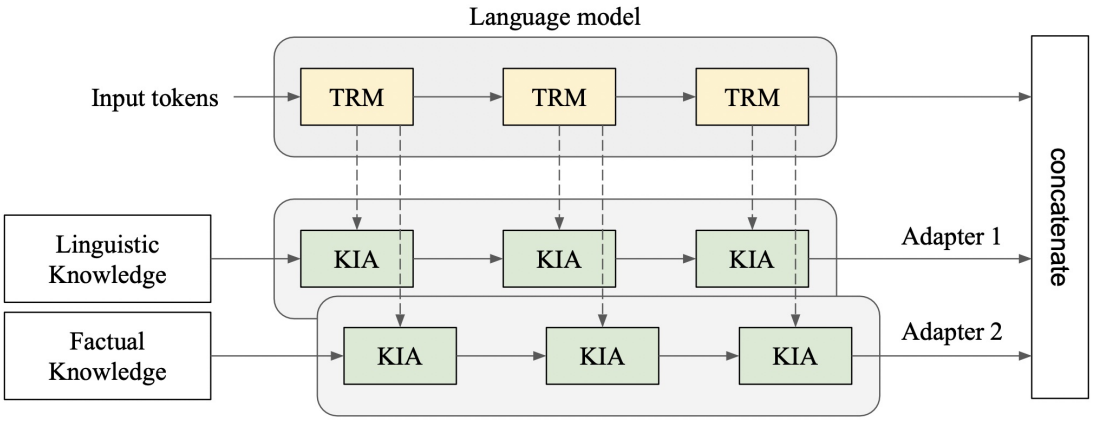

Figure 3.1: **A visual conceptualization of baseline model.** TRM represents the transformer layer while KIA represents the adapter layer

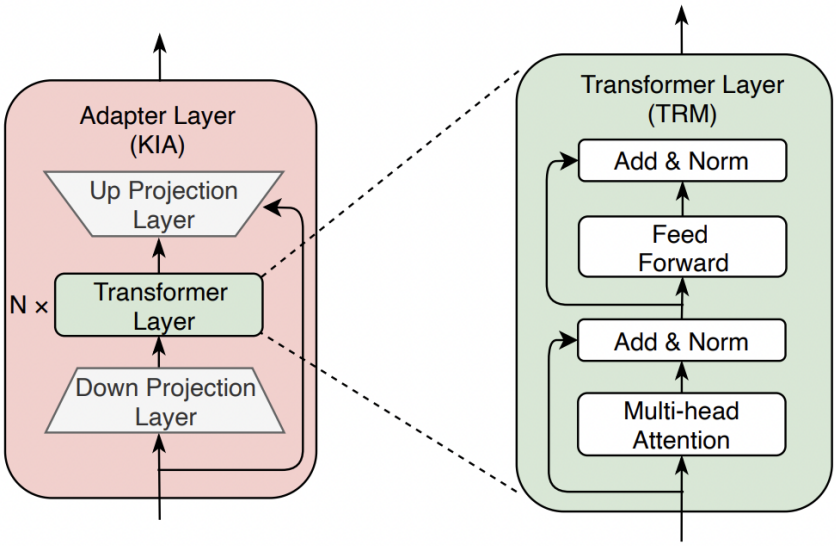

Figure 3.2: **Structure of the adapter layer.** The number of transformer layers in each adapter layer defaults to two.

Fusion among adapters via attention

In terms of knowledge fusion, we conjecture attention works better than concatenation by matching the subtle relations between knowledge and word embeddings. Input to an adapter layer is formerly a concatenation of the output hidden feature from the transformer layer and that from the former adapter layer. We now replace this concatenation with Pytorch's MultiheadAttention and set the number of heads the same as the one used in the RoBERTa-Large model (12 heads). We set the query vector as output features from the adapter, while key and value vectors are output features from the transformer layer.

Similarly, we substitute a MultiheadAttention layer for the final concatenation to produce task-specific vectors. In the situation where we have more than one adapter output vector, i.e., in the multi-knowledge case, we concatenate the two adapter vectors, pass them to a dense layer and take the output of it to attend to the PLM output.

Transfer learning with adapters

Adapters pre-trained on knowledge-specific tasks are attached to the language model at the fine-tuning stage. There are two setup options for the adapters at this stage. The first method freezes the adapters and trains only the core language model, whereas the second method trains both the adapters and language model. As there are more parameters to train, training both components should result in better results. However, this may not be the case if the adapters cannot transfer knowledge to the language model well. It is possible for the parameters of the adapters to negatively interact with those of the language model when this scenario occurs.

To study how capable the adapters and PLM are in transferring knowledge during fine-tuning, we experiment with two combinations:

$$Freeze(Adapter, PLM) = (T, F), (F, F)$$

where T means to freeze the model while F means to train it.

Contrastive sentence embeddings

Contrastive learning [41] seeks to learn effective representations by bringing semantically similar neighbors together and separating non-neighbors. Recently, Gao, Yao, and Chen [42] proposed SimCSE, an unsupervised method for learning contrastive embeddings. The training examples of SimCSE are constructed as follows. The same sentence is passed through a pre-trained encoder twice, with random dropout applied both times. This yields two distinct embeddings, which are treated as "positive pairs." Other sentences in the same batch are taken as "negatives," and the model predicts the positive one from among them. We change our PLM backbone to SimCSE and study the contribution of adopting contrastive sentence representations.

From cross-encoder to bi-encoder

A problem with our baseline implementation is that we use a cross-encoder. The maximum sequence length of the PLM is only 512, and it has to accommodate both patent texts, which are usually very long. In the PatentMatch case, approximately half of the input sequences would be truncated in various degrees when we construct them as in the next sentence prediction format. On top of that, to compare any two sentences using a cross-encoder, we need to pass the patent pair through several attention layers in the model, which is computationally more expensive than just calculating a similarity measure of the outputs in the bi-encoder scenario. A more detailed discussion on this can be found in chapter 2.

Bi-encoders (also called dual-encoders), on the other hand, compute the embedding for a given sentence only once and the embeddings can be stored in a database for quick comparison. We pass the first and second patent texts to the same encoder independently, resulting in two embeddings, A and B. These sentence embeddings can then be integrated with their corresponding knowledge embeddings and merged into one sentence embedding. The final predicted probability is obtained by computing a similarity measure between the two embeddings, such as cosine similarity. Also, bi-encoders allow for longer text inputs than cross-encoders, as the sentences

are fed into the PLM separately, rather than as a pair. We will replace the cross-encoder structure of our baseline with a bi-encoder and study its effect in the next section.

3.2.2 Ablation Study

Experimental setup

Batch Size	Epochs	Learning Rate
8	6 for PatentMatch and 4 for PatentSim	5e-6
RoBERTa-large Max Sequence Length	Adapter Hidden Size	Adapter Max Sequence Length
512	64	512

Table 3.1: **Baseline Hyperparameters**

With the baseline architecture and other design choices described in section 3.2.1, we run several experiments on PatentSim and PatentMatch to derive some insights. Table 3.1 details the model hyper-parameters and other settings of our baseline. All experiments are implemented by the Pytorch deep learning framework and ran on a single GeForce RTX 3090 GPU. See Table 3.2 for the evaluation results on PatentMatch and Table 2 for the results on PatentSim.

Results and discussion

Model Modification	Fusion in Adapter	Fusion in Final Layer	Freeze Adapter?	Freeze PLM?	Best Evaluation Accuracy
Baseline	*concat*	*concat*	*T*	*F*	***56.4%***
	concat	attention	T	F	52.0%
	attention	attention	T	F	50.0%
– w/o linAdapter	concat	concat	T	F	54.6%
	concat	concat	F	F	56.6%
– SimCSE	concat	concat	T	F	54.1%
– Bi-encoder	concat	concat	T	F	50.2%

Table 3.2: **Baseline performance and preliminary experiements on PatentMatch**

On the PatentMatch dataset, our baseline gives 56.3% accuracy, which is a 4% boost over the current best record on PatentMatch. The loss curve, however, exhibits

a rather unstable pattern that only slightly decreases after smoothing. This shows the difficulty of PatentMatch and the importance of including other patent datasets (i.e. PatentSim) for a fair evaluation of model effectiveness.

Evaluation results also indicate changing the concatenation layer to an attention layer does more harm than good for both the adapter and the final fusion, which contradicts our hypothesis. The reason can be that the attention mechanism used in fusion requires a learning rate at a different scale than the attention in PLM. Or, vanilla concatenation retains all the information while the attention selectively rules out some of it. The more we replace concatenation with the attention mechanism, the poorer our model performs.

When we remove linAdapter from our baseline, the evaluation accuracy drops to 54.6%, which shows that linAdapter indeed helps learn more knowledge and that our architecture can support continual knowledge infusion. The improvement obtained from adding linAdapter signals that it is promising for knowledge integration with adapters. In addition, making both the adapter and PLM trainable yields better results on this model version, but the additional gain in evaluation accuracy is only 0.2%. Specific to our implementation on PatentMatch, freezing the adapter but not the PLM seems to be a good tradeoff between the number of trainable parameters and the performance.

We also run an experiment that replaces the backbone RoBERTa-Large with the simple contrastive learning model SimCSE, and the performance degrades to 54.1% in accuracy. Likewise, changing the cross-encoder structure to a bi-encoder one leads to worse performance.

Model Modification	Fusion in Adapter	Fusion in Final Layer	Freeze Adapter?	Freeze PLM?	Best Evaluation AUC
Baseline	*concat*	*concat*	*T*	*F*	*90.3%*
	concat	attention	T	F	89.0%
	attention	attention	T	F	87.1%
	concat	concat	F	F	96.9%
– SimCSE	concat	concat	T	F	**98.0%**
– Bi-encoder	concat	concat	T	F	80.8%

Table 3.3: **Baseline performance and preliminary experiements on PatentMatch**

On PatentSim, our baseline produces 90.3% in terms of the AUC score. When we change the final fusion layer to attention, the AUC score decreases slightly from 90% to 89%, and the performance further degrades when the adapter layers are also attention-based. It is also noteworthy that making both the adapter and PLM trainable during fine-tuning yields a better AUC score, as high as 96.9%. This finding sheds light on the potential of training a joint objective on the PLM and adapter outputs, which we discuss in detail in section 3.3.

The best result is achieved by the version with SimCSE being the backbone model, contrary to the PatentMatch case, where the performance decreases. The inconsistency again suggests the difficulty of the PatentMatch task and the possibility that poor performance might be of the dataset instead of the model architecture.

Using a bi-encoder structure in place of the cross-encoder gives sub-optimal results, the same as the PatentMatch case. This finding also aligns with our initial assumption and the analysis from various papers. In section 3.4, we will build a more robust bi-encoder that achieves comparable results to the cross-encoder version while evading the nuisances intrinsic to cross-encoders.

Summarizing all the discoveries above, we conclude concatenation over attention, unfreezing over freezing the adapter, and cross-encoder over bi-encoder yield better outcomes on patent text comparison. Since SimCSE works well on PatentSim and slightly worse on PatentMatch, the benefit of contrastive learning might have a dependency on downstream tasks and problem formulation.

3.3 Method 2: Joint Optimization of Multiple Loss Functions

3.3.1 Methodology

Joint loss optimization is the process of optimizing a single loss function that involves multiple variables or tasks. It is frequently used in machine learning and artificial intelligence applications when the aim is to optimize multiple objectives

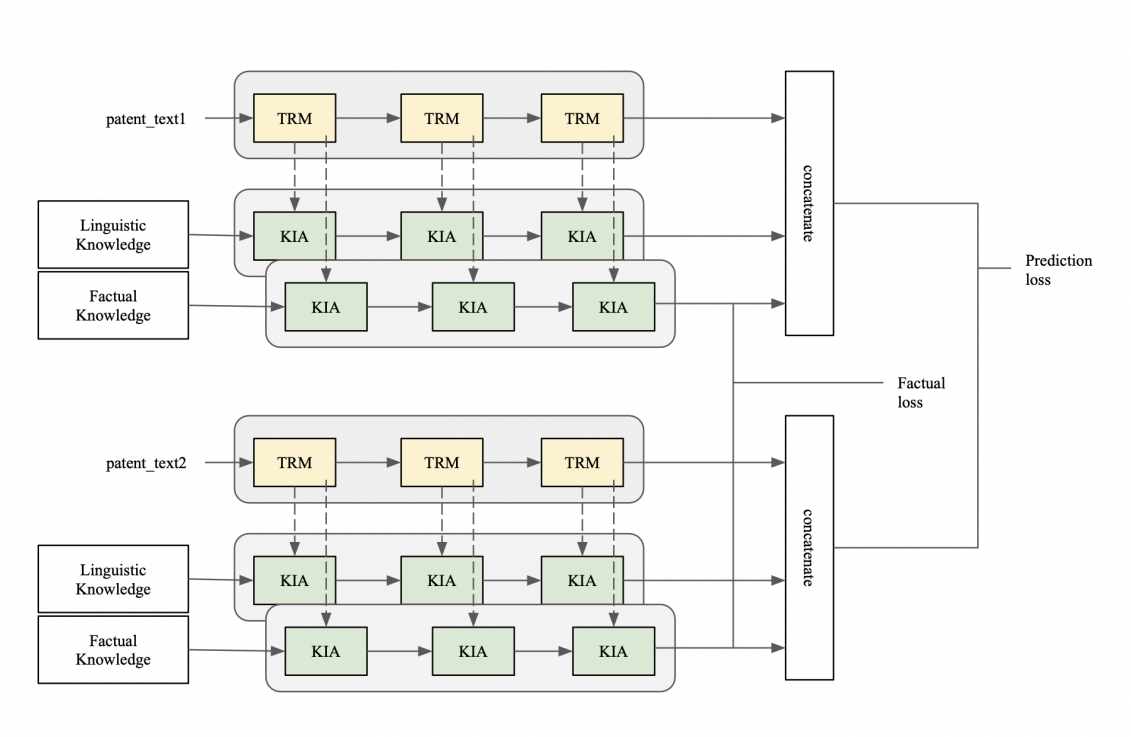

Figure 3.3: **The bi-encoder model architecture for joint optimization.** The final loss is the sum of the factual loss and the prediction loss.

simultaneously. Inspired by this idea, we add a new loss for the factual outputs to our usual prediction loss to improve the performance of the bi-encoder. This technique is similar to the multi-task learning [7] approach, except we minimize the loss across different "aspects" or "views" of the same data instead of different tasks. However, we do not introduce another loss component from the linguistic outputs as patent documents are structured alike in an academically professional style. A schematic diagram of the model architecture is as figure 3.3.

3.3.2 Experimental results and analysis

We test and analyze 6 combinations of loss functions and pooling methods: {(BCE, CLS), (BCE, MEAN), (MSE, CLS), (MSE, MEAN), (InfoNCE, CLS), (InfoNCE, MEAN)} on the PatentSim dataset for 8 epochs (225 steps). The loss functions and pooling methods of use are detailed in appendix B. The trend of loss and evaluation AUC is shown in figure 3.4 and 3.5.

The loss function gives different values according to the types of errors that it measures. Although the BCE loss is much higher than InfoNCE and MSE, this does

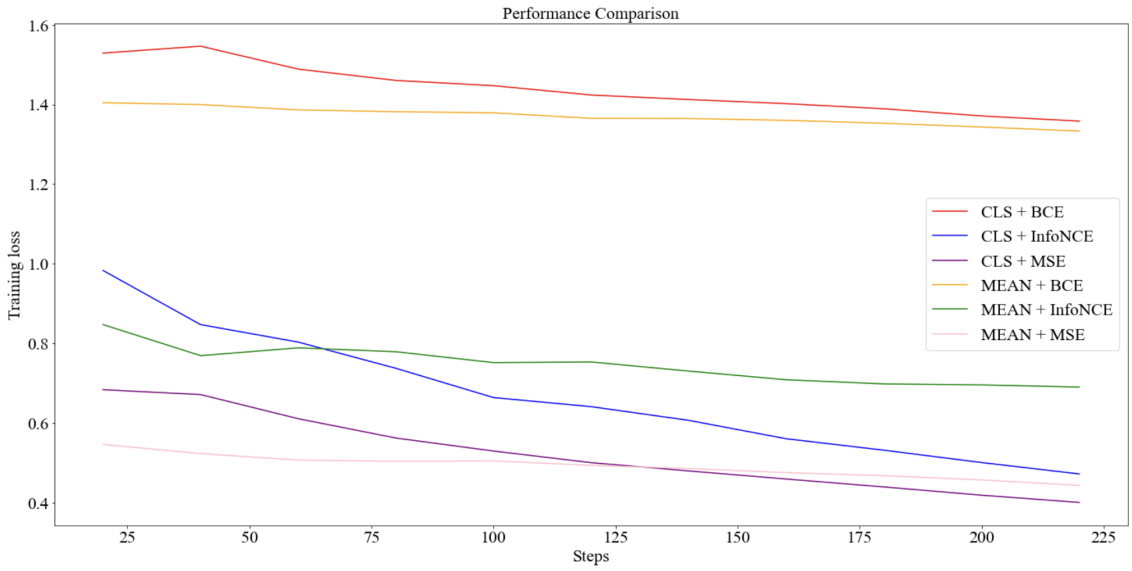

Figure 3.4: **Training loss by different combinations of loss functions and pooling methods on a bi-encoder over 8 epochs**

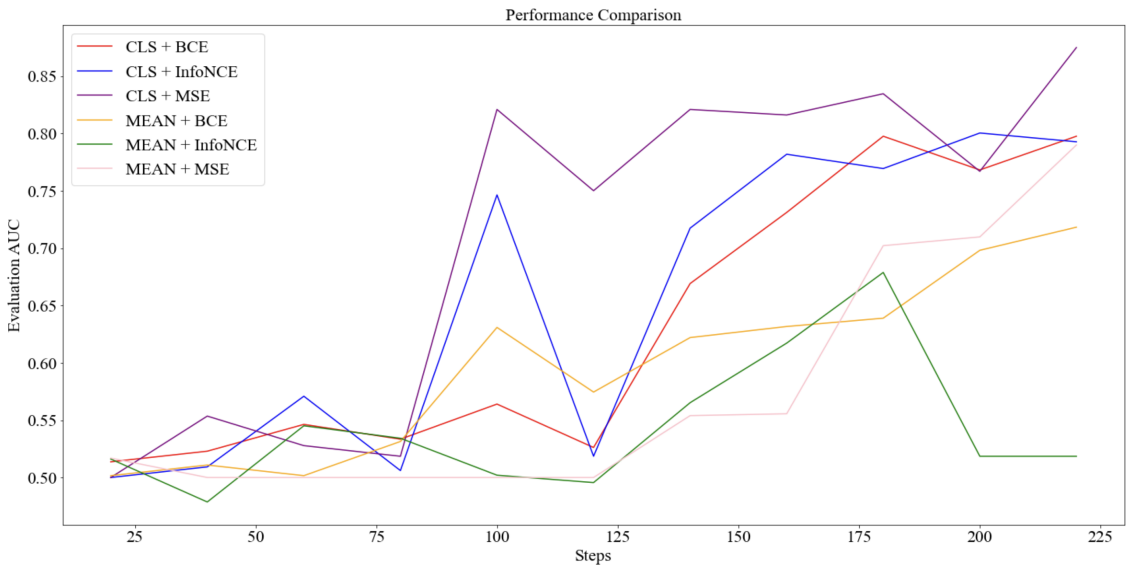

Figure 3.5: **Evaluation AUC by different combinations of loss functions and pooling methods on a bi-encoder over 8 epochs**

not imply that BCE is an inadequate choice. In contrast, the model optimized using BCE loss ranks second according to the AUC score on the evaluation set. Another observation is that CLS pooling works better than MEAN pooling, no matter which loss function we choose. To better understand the loss functions under CLS pooling, we record their evaluation results over 12 epochs in figure 3.6.

From the trend, we can tell that InfoNCE and MSE losses cease to improve evaluation results around the 300th step. The BCE loss, however, continues to enhance the performance up to the 400th step, leading to the best final AUC score.

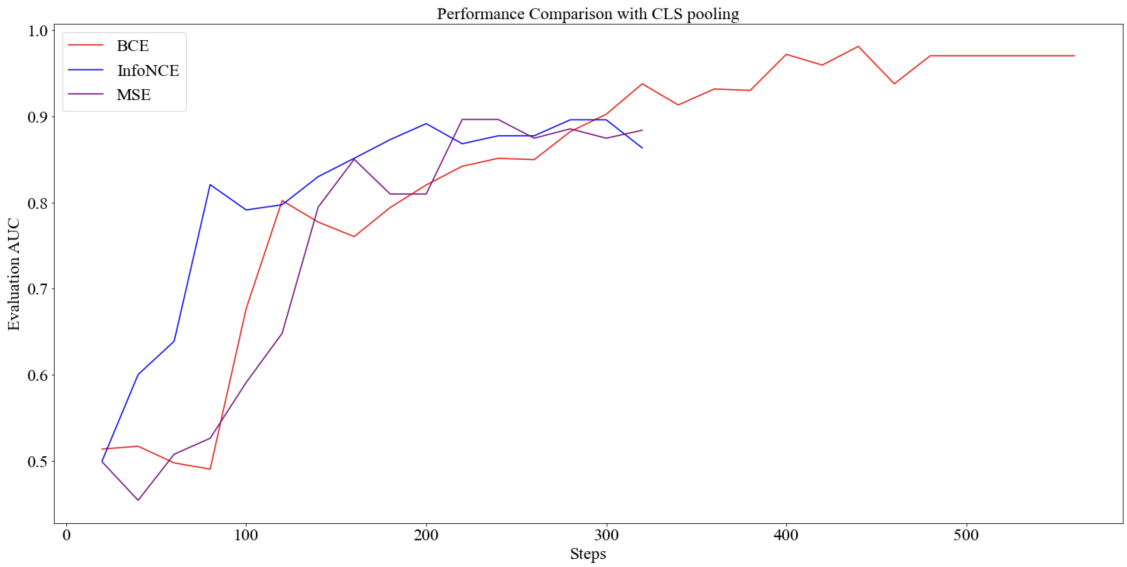

Figure 3.6: **Evaluation AUC by different loss functions under CLS pooling on a bi-encoder.** BCE is applied to 20 epochs while InfoNCE and MSE are applied to 12 epochs as they cannot improve further beyond the 12th epoch.

We also believe InfoNCE does not fully exploit its advantages due to the small batch size (batch size is 2 in our setting) limited by the computational power of the available GPU. Increasing the batch size can potentially boost the gain from contrastive learning.

In conclusion, a joint objective of the BCE loss on factual and predicted outputs reinforces the performance of the bi-encoder, with the evaluation score jumping from about 80% to over 98.1%. Indeed up to this point, we have already achieved the goal of training a robust bi-encoder comparable to its cross-encoder counterpart while having the ability of fast text retrieval, so we test our techniques on the harder dataset PatentMatch to see if the improvement still applies (figure 3.7).

Despite the success of PatentSim, the joint objective strategy combined with the bi-encoder structure still lags behind our cross-encoder baseline on PatentMatch. Worse still, the performance does not seem to improve with the decrease of training loss. We suspect the underperformance might be because of the dataset, not the model structure, so we omit PatentMatch for the succeeding experiments in chapters 3.4 and 4. Moving on, we explore knowledge distillation to refine the model even further.

Figure 3.7: **Evaluation accuracy with CLS pooling and BCE loss on a bi-encoder on PatentMatch over 20 epochs**

3.4 Method 3: Self-Distillation of Encoders

3.4.1 Methodology

Inspired by Trans-encoder [33], we strengthen our knowledgeable bi-encoder by distilling knowledge with its cross-encoder counterpart in alternation. In each round, we pseudo-label soft targets for the overall and factual outputs using an initial bi-encoder to train a cross-encoder initialized with the weights from RoBERTa-Large. Next, we load the best checkpoint of the trained cross-encoder to generate the pseudo labels for the bi-encoder and train the bi-encoder initialized from the RoBERTa-Large weights. We conduct this procedure for two rounds and discuss the results in part 3.4.2. Another design choice here is to maintain the bi- and cross-encoder weights sequentially. In other words, the weights of the cross-encoder are initialized with those of the bi-encoder which just created the pseudo labels and vice versa. Vu *et al.* [32] and Liu *et al.* [33] suggest that training sequentially leads to undesired performance most likely due to catastrophic forgetting or some error-magnifying effects. Thus we make all the knowledgeable adapters and PLMs start fresh in each training.

As advised by Vu *et al.* [32] and Liu *et al.* [33], it is crucial to start from a strong base model to have good initial pseudo labels; otherwise, the errors will be

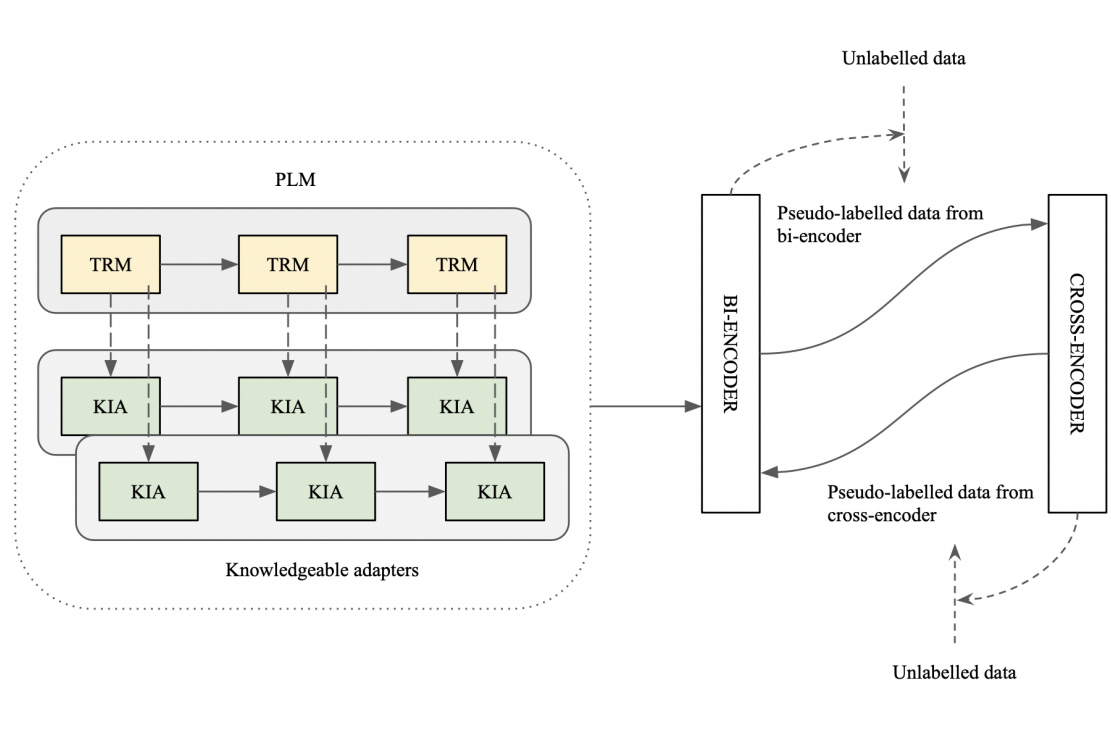

Figure 3.8: **An illustration of our self-distillation method.**

Algorithm 1 Self-distillation algorithm

Preprocess:
$\mathcal{L}$: the unlabelled set of sentence pairs
Initialize:
Initialize bi-encoder weights from figure 3.5 (CLS+Binary Cross Entropy (BCE))
$t = 0$
$w_p^0 \leftarrow w_p^*$ ▷ PLM weights
$w_a^0 \leftarrow w_a^*$ ▷ Adapter weights
repeat
$t = t + 1$
1. Generate pseudo-labels $\mathcal{R}_{cross}^t$ on $\mathcal{L}$ with the bi-encoder whose weights are given by (w_p^{t-1}, w_a^{t-1})
2. Train a cross-encoder by minimizing the joint BCE loss between $\mathcal{R}_{cross}^t$ and the predicted output logits (both overall and factual outputs)
3. Generate new pseudo-labels $\mathcal{R}_{bi}^t$ with the cross-encoder obtained above on $\mathcal{L}$
4. Train a bi-encoder by minimizing the joint BCE loss between $\mathcal{R}_{bi}^t$ and the predicted output logits (both overall and factual outputs.)
The resulting bi-encoder has new weights given by (w_p^t, w_a^t)
until the maximum number of iterations is reached

propagated through incorrect data. As a result, we start with our best bi-encoder checkpoint, trained through joint optimization with the bce loss over 20 epochs. See figure 3.8 for a brief paradigm of our self-distillation method and algorithm 1 for the exact procedure.

3.4.2 Experimental results and analysis

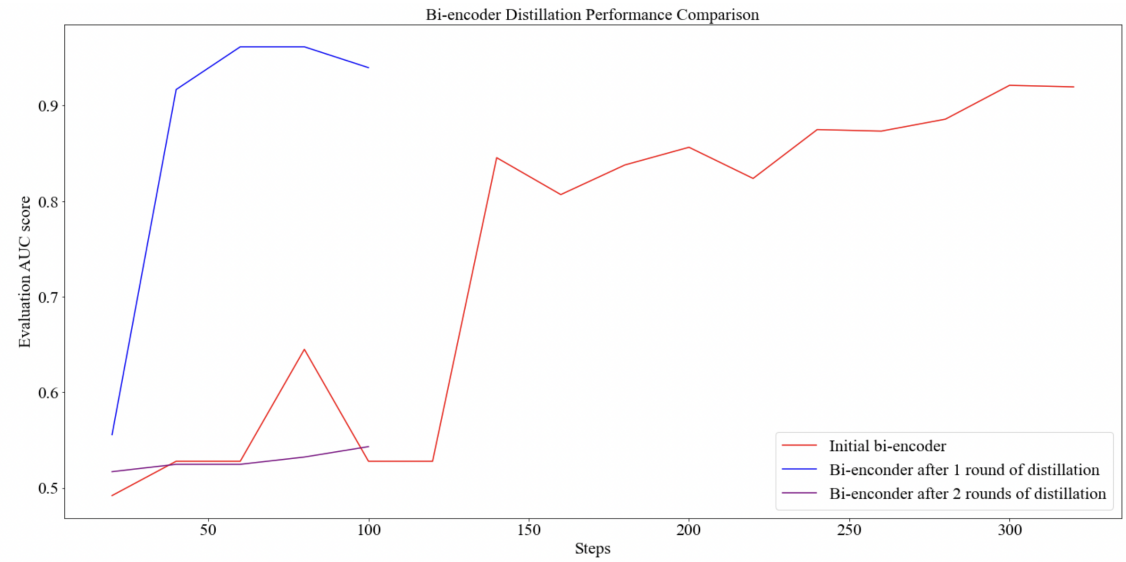

Figure 3.9: **Bi-encoder Distillation Performance Comparison.** Except the initial bi-encoder is trained over 20 epochs, the other two self-distilled bi-encoders are trained over 2 epochs. The training trend of the initial bi-encoder is only shown till the 12th epoch for better rendering.

Figure 3.10: **Cross-encoder Distillation Performance Comparison.** Except the initial bi-encoder is trained over 4 epochs, the other two self-distilled cross-encoders are trained over 3 epochs. The training trend of the vanilla bi-encoder is only shown till the 3th epoch for better rendering.

In this section, we investigate the effectiveness of self-distillation on the PatentSim dataset by comparing the performance of the newly obtained bi- and cross-encoder with their previous versions, respectively. We conduct the distillation procedure in two iterations.

For the bi-encoder, one iteration should suffice to leverage the advantage of self-distillation. Distilling with more than one iteration results in overfitting the pseudo-labels and downgrading the evaluation efficiency. As seen in figure 3.9, the bi-encoder's convergence after one iteration of self-distillation is the fastest, while the one after two iterations cannot converge at all. The initial bi-encoder denoted in figure 3.9 is the starting checkpoint of the bi-encoder in our experiment, which is pre-trained with a joint BCE loss over 20 epochs in section 3.3. Though the initial bi-encoder attains the highest validation AUC score (98.1%) after 20 epochs, the one after one round of self-distillation achieves similar AUC score (96%) within just 2 epochs.

Likewise, distilling for one iteration is enough to obtain a strong cross-encoder. Despite a reduction in training loss, overfitting is even more severe after two iterations, as the evaluation AUC score remains unchanged. The new cross-encoder also converges much more quickly than its vanilla version. However, their difference is not as significant as in the bi-encoder case, which we can attribute to the natural characteristics of cross-encoders to converge faster. Such quick convergence of the bi-encoder opens the door to possibly transforming it into a few-shot retriever, and we will delve deeper into retrieval in the following chapter 4.

Chapter 4

K-PAR: The Knowledgeable Prior Art Retriever

We propose K-PAR, the knowledgeable prior art retriever based on the bi-encoder model structure. In the following sections, we will discuss our methods for end-to-end retrieval and the evaluation results.

4.1 Methodology

Converting a bi-encoder trained on patent pairs to a retriever is intuitive: we first construct the embedding pool of a range of prior art we want to search from, get the embedding of the new patent description, and compare it against the whole embedding pool to retrieve the relevant prior art with the highest rankings.

We employ representation-based similarity as the interaction scheme in our experiment as we assume the complexity of the RoBERTa-Large model produces embeddings that already strongly characterize the patent texts. We choose cosine similarity implemented with PyTorch as the similarity measure.

4.2 Experimental setup

4.2.1 Test set formulation

Our previous formulation of PatentSim extracts 1000 patent pairs, from which 900 are for training and the others are for validation. The test dataset for retrieval consists of 100 patent text extracted from the remaining corpus. To sample the test data, we make sure the ground truth patent to each test patent text does not also reside in the test set. In other words, if patent A is the ground truth relevant to patent B and patent B is in the test set, patent A should be in the retrievable database.

Next, we rank all patent text according to the number of their ground truth in the database in descending order. The first 100 patents with the most ground truth in our database become the test set. The most cited patents in the selected test set have as many as eighteen ground truths, and the least cited ones have two ground truths.

4.2.2 Evaluation metrics

There are three ranking-related metrics (hit ratio, MRR@k, and mAP@k) to evaluate our retrieval system K-PAR. Hit ratio is simply the fraction of query patents for which the relevant prior arts are included in the retrieved set. The Mean Reciprocal Rank (MRR) [43] only looks at the one top relevant item in each retrieved list, and averages the reciprocal of the rank among the whole query set. For instance, if the first correct item for a query is at rank 3, the RR score would be 1/3. The Mean Average Precision (mAP) [44] calculates sliding precisions for the first l (l = 0,1,2,..,k) retrieval. Take k=5 and assume the correct items are at ranks 3 and 5. In this case, we have precision@5 = [0, 0, 1/3, 1/4, 2/5], and the average precision would be (1/3+1/4+2/5)/5 = 0.1967. Here we do not consider ranking with weightings as the relative importance of each relevant item is not given in the dataset.

4.2.3 Indexing and quering

Computing and storing prior art embeddings are straightforward. We pass each patent document to our bi-encoder checkpoint and concatenate their output vectors to form the embedding pool, saved to disk as a Numpy object. This indexing is offline, casting no additional overheads in query time. We maintain a mapping from each embedding to its document of origin for referencing the patent documents in text format.

When querying a given patent description, we use a two-stage procedure to retrieve the most relevant prior art from the entire database. The first stage aims at approximating and filtering out clusters closer to the target query in vector space. The second stage is responsible for fine-graining the most similar embeddings in those clusters by matching them with the query embedding one by one.

The approximation stage narrows our probe space via the off-the-shelf similarity search library Faiss[1] [45]. We use IVF (inverted file index) to map each prior art embedding to their respective clusters. Similar to K-means clustering [46], an embedding belongs to the partition with the closest centroid, and we choose the number of clusters to be the square root of our samples from empirical knowledge. Next, Faiss compares the query embedding with all the centroid embeddings, and we take the nearest N = 3 clusters. To better ranking quality, embeddings assigned to these clusters will be re-ranked exhaustively to form the final ranking in the fine-graining stage. As accurate matching is essential in patent examination, we do not apply more advanced approximation techniques (e.g. product quantization), which trade less accuracy for quicker inference.

4.3 Experimental results and analysis

The results in terms of six different metrics are tabulated in Table 4.1, and a full-text retrieved example is recorded in Table 4.3.

We observe that the (pre-trained model, adapter) checkpoints from training the

[1]Faiss is developed by Facebook AI and is publicly available at `https://github.com/facebookresearch/faiss`

Model Checkpoint	Index	HitRatio@5	MRR@5	MAP@5
Bi-encoder				
– vanilla	Flat	**0.126**	**0.201**	**0.121**
	IVF	0.067	0.148	0.077
– self-distillation	Flat	0.015	0.074	0.034
	IVF	0.03	0.111	0.053
Cross-encoder				
– vanilla	Flat	0.081	0.119	0.07
	IVF	0.03	0.041	0.016
– self-distillation	Flat	0.096	0.129	0.081
	IVF	0.074	0.142	0.08
– SimCSE	Flat	0.089	0.181	0.105
	IVF	0.059	0.148	0.081

Model Checkpoint	Index	HitRatio@10	MRR@10	MAP@10
Bi-encoder				
– vanilla	Flat	0.267	0.333	0.24
	IVF	0.033	0.083	0.037
– self-distillation	Flat	0	0	0
	IVF	0.067	0.095	0.032
Cross-encoder				
– vanilla	Flat	**0.3**	0.4	**0.278**
	IVF	0.067	0.222	0.095
– self-distillation	Flat	0.133	0.333	0.202
	IVF	0.067	**0.667**	0.195
– SimCSE	Flat	0.067	0.333	0.129
	IVF	0.067	0.333	0.129

Table 4.1: **Retrieval evaluation results of bi-encoders** initialized with different (pre-trained model, adapters) checkpoint pairs saved in our implementations in chapters 3 to 3.4

vanilla bi-encoder retrieve most accurately the top-5 prior art relevant to a patent application, followed by checkpoints from cross-encoder trained with a SimCSE backbone. This finding is reasonable since the model structure we use in sentence pair modeling is the same as in retrieval. In other words, the bi-encoder checkpoints are more likely to adapt well to retrieval tasks that use a bi-encoder structure. By comparing the initialization of the retriever with different variations of cross-encoder checkpoints, we see that contrastive learning improves retrieval performance. We attribute the effect of contrastive learning to the fact that it learns better dense sentence representations that are useful in comparing text embeddings for retrieval.

While self-distillation reduces training time and increases evaluation accuracy on patent similarity tasks, it degrades the model on retrieval tasks. For cross-encoders,

self-distillation only slightly improves the retrieval performance, but it harms the retrieval ability of bi-encoders much more significantly. Our interpretation is that self-distillation produces models that are experts in specific tasks, so it is less resilient for them to adapt to other similar but related ones.

We use Faiss' indexing tool (flat and inverted file index) for efficient similarity searches. At search time, a flat index decodes all the indexed vectors of the prior art and compares them with the vector of the query patent. An inverted file index (IVF) adds the centroids to the flat index, as described in section 4.2.3. Since IVF approximates the exact search by first bundling search vectors into groups identified by their centroids, its retrieval results are subject to randomness in the clustering methods and may differ in each run. Overall, the flat index is more accurate than IVF, but IVF is faster in retrieval. On 1000 query patents, the flat index consumes 0.68 to 0.72 seconds to retrieve, while IVF only uses 0.01 to 0.02 seconds. In the context of prior art retrieval, the scale of time improvement of IVF is not enough to trade over better accuracy implied by a flat index. Also, most time used in bi-encoder retrieval is due to loading the pre-computed prior art embeddings, which is only around four seconds. As a result, we conclude a flat index is a more reasonable option for our case.

We also conduct retrieval experiments using a cross-encoder structure initialized by the (pre-trained model, adapter) checkpoints from the vanilla, self-distillation, and SimCSE cross-encoder training (see table 4.2). Opposite to our guess, the retrieval outcomes are unsatisfactory. This could be due to severe overfitting, where the model fails to handle any out-of-domain comparison. To make matters worse, every pair of prior art and patent application must undergo heavy computation within the transformer to generate a similarity score, increasing the retrieval time from a few seconds to 70 to 80 minutes.

Model Checkpoint	HitRatio@5	MRR@5	MAP@5
Cross-encoder			
– vanilla	0.0074	0.0093	0.0033
– self-distillation	0.0074	0.0185	**0.0095**
– SimCSE	**0.022**	**0.024**	0.006

Table 4.2: **Retrieval evaluation results of cross-encoders** initialized with different (pre-trained model, adapters) checkpoint pairs saved in our implementations in chapters 3 to 3.4

Patent application
Systems, assemblies, and methods for treating a bronchial tree Systems, assemblies, and methods to treat pulmonary diseases are used to decrease nervous system input to distal regions of the bronchial tree within the lungs. Treatment systems damage nerve tissue to temporarily or permanently decrease nervous system input. The treatment systems are capable of heating nerve tissue, cooling the nerve tissue, delivering a flowable substance that cause trauma to the nerve tissue, puncturing the nerve tissue, tearing the nerve tissue, cutting the nerve tissue, applying pressure to the nerve tissue, applying ultrasound to the nerve tissue, applying ionizing radiation to the nerve tissue, disrupting cell membranes of nerve tissue with electrical energy, or delivering long acting nerve blocking chemicals to the nerve tissue.
Correctly retrieved prior art (at rank 3)
Devices for creating collateral in the lungs The devices and methods disclosed herein are directed to altering gaseous flow within a lung to improve the expiration cycle of, for instance, an individual having Chronic Obstructive Pulmonary Disease. More particularly, these devices and methods produce and to maintain collateral openings or channels through the airway wall so that expired air is able to pass directly out of the lung tissue to facilitate both the exchange of oxygen ultimately into the blood and/or to decompress hyper-inflated lungs. The devices and methods also disclose locating and selecting a site for creation of a collateral opening.
Correctly retrieved prior art (at rank 4)
Bronchial stenter A device and method for treating collapsed bronchial tubes found in patients with chronic obstructive pulmonary disease and asthma are provided. The device which has a balloon at a distal end delivers energy so that the tissue is indirectly, i.e., conductively, heated by contact with a heated surface of the balloon which has been in turn heated by a heated fluid. The method includes heating the bronchial tube to cause at least a portion of the cross links of the collagen in the wall to unlink/open and subsequently form new cross links after the collagen fibers have realigned. The procedure effectively reinforces the structural integrity of the wall and thereby prevents the lumen from collapsing.

Table 4.3: **A full-text example of prior art retrieval**

Chapter 5

Future Work

5.1 Open-domain question answering

It is possible to extend the prior art retriever to answer open questions related to patent texts. Modern OpenQA models [47] use the "retriever-reader" architecture. The retriever extracts relevant documents for a given query. These documents, along with the query, are then passed to a reader to generate the answer. An application of this will be fact-checking and patent information verification on the patent application. We encourage future efforts in adapting our prior art retriever to a patent fact checker.

5.2 Knowledgeable adapter pre-training

The factual adapter used in our patent model is pre-trained on the T-rex dataset. Instead of a general relation classification task as such, we hypothesize pre-training an adapter on patent-specific data can further enhance our model's performance. One possible choice of dataset is the Google Patent Phrase Similarity Dataset [48]. It consists of human-rated contextual phrase-to-phrase matchings centering around technical terms from patents, taking on twelve similarity levels. Another proposal is to pre-train the factual adapter on PatentSim, which can potentially improve performance on the more difficult dataset, PatentMatch.

5.3 Memory-efficient storage

As in the case of all retrieval systems, they must maintain high retrieval quality while minimizing inference time and space footprints. We have discussed the balancing of computational speedup and the performance of IVF in section 4.2.3. However, we did not explore the storage efficiency of incorporating approximated indices such as IVF. Our current implementation creates large memory overheads since it directly stores the uncompressed embeddings on disk. A feasible solution is to represent each embedding as a centroid and its residual vector to that centroid [49]. As a result, we only need to store the centroids' embeddings and the residual vectors, which are usually small in size. This method of approximation can compromise retrieval accuracy. We leave the study of the trade-off between retrieval performance and memory efficiency for future work.

5.4 Human-in-the-loop for continual improvement

The prior art database used by patent offices is not fixed over time but constantly expanding to include the newest approved patents. Consequently, our retrieval model should be able to extend its coverage to such new patents. A possible approach is to have human patent examiners validate the retrieval outcomes and update the model on whether the retrieved results are adequate. In this method, the correctly retrieved prior art and the new patent form positive examples, while the incorrectly retrieved pairs constitute the negative examples. Once we collect enough of them, we can train a new bi-encoder using a similar self-distillation strategy described in chapter 3.4, with the previous bi-encoder pseudo-labeling the training data for the new bi-encoder. As training with self-distillation allows for few-shot learning, it is suitable for training on small data sizes in cases like patent data. Nevertheless, self-distilled models might become overly specialized in their original tasks and not easily convert to good retrievers. The challenge remains in infusing knowledge from new training examples without retraining the model from scratch, so we urge further research on this topic to scale our patent retriever to actual use.

Chapter 6

Conclusion

Overall, we show that adapter-based knowledge integration is feasible and appropriate for domain-specific patent data. A baseline is successfully implemented and surpasses the current state-of-the-art models by a large margin. We also discuss design choices of the baseline through an ablation study. On top of that, three model architectures are tested and analyzed: the bi-encoder Siamese structure, the joint loss objective, and self-distillation. We demonstrate these architectural changes can improve our baseline model for higher evaluation accuracy or faster time inference.

In the final portion of our work, we present a novel prior art retriever K-PAR. As K-PAR directly takes our pre-trained knowledgeable model on patent pair modeling off the shelf, we suggest future work to optimize this basic retrieval setup and the end-to-end retrieval process. This project hopes to contribute to the rising field of document retrieval in the patent context. We will continue investigating the retrieval part to scale our work further.

Reflection on Learning Outcome Attainment

Through this final year project, I have gained invaluable experience and knowledge that will later be useful in my career development. Firstly, I have sharpened my ability to carry out independent research. Unlike in the classroom, I encounter never-before-seen complex problems and find novel solutions to them creatively. I also learn to think logically and critically about the model design and implementation, which enhance my problem-solving skills.

Secondly, I have also grown my communication skill through regular updates on my work to my supervisor and lab scientists. I must explain myself clearly and academically to convey my ideas in exchange for helpful opinions. Such practice prepares me to communicate project updates in the workplace and makes me a more competitive presenter.

This project is also a good opportunity for me to develop time management and the skills to plan and prioritize tasks. Consistent and adequate commitment is crucial to produce quality work, especially since FYP is a one-year project that takes time to accumulate. Overall, it is a challenging yet fruitful journey for me, and the experience I learn from it is enormous.

Appendix A

Software Environment and Required Packages

Our work is publicly available on Github.

Software environment:

- Python version: 3.9.13
- Conda version: 22.9.0
- Cuda version: 11.2
- Git version: 2.17.1
- GPU version: GeForce RTX 3090

Required packages:

- info-nce-pytorch 0.1.4
- scikit-learn 0.24.2
- huggingface-hub 0.4.0
- transformers 4.18.0
- sentencepiece 0.1.97
- sqlalchemy 1.3.13
- sqlite 1.30.2
- torchvision 0.9.0+cu111
- torch: 1.3.1
- pandas: 1.1.5
- numpy: 1.16.2
- faiss: 1.7.0
- tensorflow: 2.6.2
- tensorboard: 2.6.0

Appendix B

Loss functions and pooling methods for joint optimization

In this section, we will describe the loss functions and pooling methods that we used in our experiments to train a joint objective.

B.0.1 Binary cross entropy

The BCE loss is a widely-used loss function in machine learning for binary classification tasks. This function measures the distance between the predicted probabilities and the ground truth, typically represented as 0 and 1. It is defined as follows:

$$\text{BCE} = -\frac{1}{N}\sum_{i=0}^{N} y_i \cdot \log\left(\hat{y}_i\right) + \left(1 - y_i\right) \cdot \log\left(1 - \hat{y}_i\right)$$

where y_i represents the true binary label, and $\hat{y}$ represents the predicted probability of the positive class.

When y_i equals 1, the loss function reduces to $-\log(\hat{y})$, meaning that the loss will decrease with a higher predicted probability. On the other hand, the loss function reduces to $-\log(1-\hat{y})$ when y_i equals 0, which means the loss will increase as the predicted probability increases. In other words, the binary entropy loss function punishes the model more severely for making incorrect predictions with high confidence and less for those with low confidence.

In our experiment, we apply the BCE loss to minimize the KL divergence of cosine similarity between the final outputs and the actual labels. We combine the loss emitted from the factual and prediction outputs as the final loss. Namely,

$$FinalLoss = \mathrm{BCE}(\cos(\mathrm{output1}_{\mathrm{fac}}, \mathrm{output2}_{\mathrm{fac}})) + \mathrm{BCE}(\cos(\mathrm{output1}, \mathrm{output2}))$$

B.0.2 Mean squared error

The Mean Squared Error (MSE) loss is typically used for regression tasks. However, we include it as one of our choices of loss functions because the prediction outcome is a real value (a probability between 0 and 1) instead of hard labels represented as integers. Its mathematical form is given by

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Same as BCE, we apply the MSE loss to minimize the distance between the cosine similarities and the actual labels. The final loss is the sum of the factual and prediction losses.

B.0.3 InfoNCE

The InfoNCE loss [50] uses categorical cross-entropy to distinguish the positive sample from a group of unrelated noise samples. It applies contrastive learning, which increases similarities between data of the same class and decreases similarities between those from different classes. This ensures that data with similar meanings are situated closer in the embedding space while those with different meanings are further apart.

For a given set of parameters θ, the positive sample should be drawn from the conditional distribution $p(x|\theta)$, while N-1 negative samples are drawn from the proposed distribution $p(x)$, independent of the θ. The probability of correctly detecting the positive sample is:

$$\frac{f(\mathbf{x_{pos}}, \theta)}{\sum_{\mathbf{x}' \in X} f(\mathbf{x}', \theta)}$$

Maximizing the log probability of the above across all samples in each batch is equivalent to minimizing

$$\text{InfoNCE} = -\mathbb{E}\left[\log \frac{f(\mathbf{x_{pos}}, \theta)}{\sum_{\mathbf{x}' \in X} f\left(\mathbf{x}', \theta\right)}\right]$$

For every patent document pair (patent1, patent2), the negative samples concerning a particular patent1 are all the other instances of patent2 in the same batch if the ground truth is 1, while if the ground truth is 0, the negative samples are its corresponding patent2 plus all except for one of the other patent2 randomly selected in the same batch. This leave-one-out option is to ensure the dimensions for the chosen negative samples match across all examples of patent1.

The positive sample corresponding to each patent1 is just patent2 if the ground truth is 1. Otherwise, we directly set patent1 as the positive sample if the ground truth is 0. By right, patent1 should be passed down to the encoder twice to generate a slightly different embedding due to the dropout layers in the encoder. Though this design choice can better facilitate contrastive learning, it incurs computational overheads on top of the already strained resources used for bi-encoder training (as we need to train two parallel models at the same time.) Therefore, we leave this direction of exploration as future work.

We construct negative and positive samples in a similar way to calculate the factual loss. Same as BCE and MSE, the final loss is the sum of the factual loss and prediction loss.

B.0.4 CLS and MEAN pooling

We consider two pooling methods (CLS and MEAN) to downsample the output vectors into features for cosine similarity measurement. CLS pooling takes the CLS token embedding as the feature vector, while mean pooling averages token-level embeddings element-wise.

Bibliography

[1] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *ArXiv*, vol. abs/1810.04805, 2019.

[3] T. Brown, B. Mann, N. Ryder, *et al.*, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., 2020, pp. 1877–1901. [Online]. Available: `https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf`.

[4] R. Wang, D. Tang, N. Duan, *et al.*, "K-Adapter: Infusing Knowledge into Pre-Trained Models with Adapters," in *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, Association for Computational Linguistics, Aug. 2021, pp. 1405–1418. DOI: `10.18653/v1/2021.findings-acl.121`.

[5] P. He, X. Liu, J. Gao, and W. Chen, "Deberta: Decoding-enhanced bert with disentangled attention," in *International Conference on Learning Representations*, 2021. [Online]. Available: `https://openreview.net/forum?id=XPZIaotutsD`.

[6] D. Chicco, "Siamese neural networks: An overview," *Methods in molecular biology (Clifton, N.J.)*, vol. 2190, pp. 73–94, 2021, ISSN: 1064-3745. DOI: `10.1007/978-1-0716-0826-5_3`. [Online]. Available: `https://doi.org/10.1007/978-1-0716-0826-5_3`.

[7] R. Caruana, "Multitask learning," *Machine Learning*, vol. 28, Jul. 1997. DOI: `10.1023/A:1007379606734`.

[8] X. Liu, P. He, W. Chen, and J. Gao, "Multi-task deep neural networks for natural language understanding," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 4487–4496. DOI: `10.18653/v1/P19-1441`. [Online]. Available: `https://aclanthology.org/P19-1441`.

[9] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019.

[10] T. Schick and H. Schütze, "Exploiting cloze-questions for few-shot text classification and natural language inference," in *Conference of the European Chapter of the Association for Computational Linguistics*, 2020.

[11] M. Marelli, S. Menini, M. Baroni, L. Bentivogli, R. Bernardi, and R. Zamparelli, *The SICK (Sentences Involving Compositional Knowledge) dataset for relatedness and entailment*, Zenodo, May 2014. DOI: 10.5281/zenodo.2787612. [Online]. Available: https://doi.org/10.5281/zenodo.2787612.

[12] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, "A large annotated corpus for learning natural language inference," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics, 2015.

[13] T. Khot, A. Sabharwal, and P. Clark, "Scitail: A textual entailment dataset from science question answering," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018. DOI: 10.1609/aaai.v32i1.12022. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/12022.

[14] A. Williams, N. Nangia, and S. Bowman, "A broad-coverage challenge corpus for sentence understanding through inference," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, New Orleans, Louisiana: Association for Computational Linguistics, 2018, pp. 1112–1122. [Online]. Available: http://aclweb.org/anthology/N18-1101.

[15] J. Thorne, A. Vlachos, O. Cocarascu, C. Christodoulopoulos, and A. Mittal, "The fact extraction and VERification (FEVER) shared task," in *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 1–9. DOI: 10.18653/v1/W18-5501. [Online]. Available: https://aclanthology.org/W18-5501.

[16] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman, "GLUE: A multi-task benchmark and analysis platform for natural language understanding," in *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 353–355. DOI: 10.18653/v1/W18-5446. [Online]. Available: https://aclanthology.org/W18-5446.

[17] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ questions for machine comprehension of text," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 2383–2392. DOI: 10.18653/v1/D16-1264. [Online]. Available: https://aclanthology.org/D16-1264.

[18] A. Wang, Y. Pruksachatkun, N. Nangia, *et al.*, "Superglue: A stickier benchmark for general-purpose language understanding systems," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32, Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/4496bf24afe7fab6f046bf4923da8de6-Paper.pdf.

[19] Y. Nie, A. Williams, E. Dinan, M. Bansal, J. Weston, and D. Kiela, "Adversarial NLI: A new benchmark for natural language understanding," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, 2020.

[20] P. Colon-Hernandez, C. Havasi, J. B. Alonso, M. Huggins, and C. Breazeal, "Combining pre-trained language models and structured knowledge," *CoRR*, vol. abs/2101.12294, 2021. arXiv: 2101.12294 [cs.CL].

[21] Z. Zhang, X. Han, Z. Liu, X. Jiang, M. Sun, and Q. Liu, "ERNIE: Enhanced language representation with informative entities," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 1441–1451. DOI: 10.18653/v1/P19-1139.

[22] L. Weijie, Z. Peng, Z. Zhe, *et al.*, "K-BERT: Enabling language representation with knowledge graph," in *Proceedings of AAAI 2020*, 2020.

[23] O. Agarwal, H. Ge, S. Shakeri, and R. Al-Rfou, "Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, Jun. 2021, pp. 3554–3565. DOI: 10.18653/v1/2021.naacl-main.278.

[24] R. Xie, Z. Liu, J. Jia, H. Luan, and M. Sun, "Representation learning of knowledge graphs with entity descriptions," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, 2016.

[25] X. Wang, T. Gao, Z. Zhu, *et al.*, "Kepler: A unified model for knowledge embedding and pre-trained language representation," *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 176–194, 2021. DOI: 10.1162/tacl_a_00360.

[26] R. Logan, N. F. Liu, M. E. Peters, M. Gardner, and S. Singh, "Barack's wife hillary: Using knowledge graphs for fact-aware language modeling," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 5962–5971. DOI: 10.18653/v1/P19-1598.

[27] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. DOI: 10.1162/neco.1997.9.8.1735.

[28] M. E. Peters, M. Neumann, R. L. Logan, *et al.*, "Knowledge enhanced contextual word representations," in *EMNLP*, 2019.

[29] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M.-W. Chang, "Realm: Retrieval-augmented language model pre-training," in *Proceedings of the 37th International Conference on Machine Learning*, ser. ICML'20, JMLR.org, 2020.

[30] H. Neil, G. Andrei, J. Stanislaw, *et al.*, "Parameter-efficient transfer learning for nlp," in *ICML*, 2019.

[31] G. Hinton, J. Dean, and O. Vinyals, "Distilling the knowledge in a neural network," Mar. 2014, pp. 1–9.

[32] T. Vu, M.-T. Luong, Q. V. Le, G. Simon, and M. Iyyer, "Strata: Self-training with task augmentation for better few-shot learning," in *Conference on Empirical Methods in Natural Language Processing*, 2021.

[33] F. Liu, S. Havrylov, Y. Jiao, J. Massiah, and E. Yilmaz, "Trans-encoder: Unsupervised sentence-pair modelling through self- and mutual-distillations," *ArXiv*, vol. abs/2109.13059, 2021.

[34] Y. Lu, Y. Liu, J. Liu, *et al.*, "Ernie-search: Bridging cross-encoder with dual-encoder via self on-the-fly distillation for dense passage retrieval," *ArXiv*, vol. abs/2205.09153, 2022.

[35] O. Khattab and M. Zaharia, "Colbert: Efficient and effective passage search via contextualized late interaction over BERT," *CoRR*, vol. abs/2004.12832, 2020. arXiv: 2004.12832. [Online]. Available: https://arxiv.org/abs/2004.12832.

[36] Y. LeCun, B. Boser, J. Denker, *et al.*, "Handwritten digit recognition with a back-propagation network," in *Advances in Neural Information Processing Systems*, vol. 2, Morgan-Kaufmann, 1989.

[37] L. Helmers, F. Horn, F. Biegler, T. Oppermann, and K.-R. Müller, "Automating the search for a patent's prior art with a full text similarity search," *PLoS ONE*, vol. 14, no. 3, e0212103, 2019.

[38] J. Risch, N. Alder, C. Hewel, and R. Krestel, "PatentMatch: A dataset for matching patent claims with prior art," *ArXiv e-prints*, 2020. arXiv: 2012.13919 [cs.CL].

[39] H. ElSahar, P. Vougiouklis, A. Remaci, *et al.*, "T-rex: A large scale alignment of natural language with knowledge base triples," in *International Conference on Language Resources and Evaluation*, 2018.

[40] Y. Zhu, R. Kiros, R. Zemel, *et al.*, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," in *The IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015.

[41] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, 2006, pp. 1735–1742. DOI: 10.1109/CVPR.2006.100.

[42] T. Gao, X. Yao, and D. Chen, "SimCSE: Simple contrastive learning of sentence embeddings," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 6894–6910. DOI: 10.18653/v1/2021.emnlp-main.552. [Online]. Available: https://aclanthology.org/2021.emnlp-main.552.

[43] N. Craswell, "Encyclopedia of database systems," in L. Liu and M. T. Özsu, Eds. Springer US, 2009, ch. Mean Reciprocal Rank, pp. 1703–1703.

[44] O. Vechtomova, "Book review: Introduction to information retrieval by christopher D. manning, prabhakar raghavan, and hinrich Schütze," *Computational Linguistics*, vol. 35, no. 2, Jun. 2009. DOI: 10.1162/coli.2009.35.2.307. [Online]. Available: https://aclanthology.org/J09-2006.

[45] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2019.

[46] J. MacQueen, "Classification and analysis of multivariate observations," in *5th Berkeley Symp. Math. Statist. Probability*, University of California Los Angeles LA USA, 1967, pp. 281–297.

[47] F. Zhu, W. Lei, C. Wang, J. Zheng, S. Poria, and T. Chua, "Retrieving and reading: A comprehensive survey on open-domain question answering," *CoRR*, vol. abs/2101.00774, 2021. arXiv: 2101.00774. [Online]. Available: https://arxiv.org/abs/2101.00774.

[48] G. Aslanyan and I. Wetherbee, "Patents phrase to phrase semantic matching dataset," 2022.

[49] K. Santhanam, O. Khattab, J. Saad-Falcon, C. Potts, and M. A. Zaharia, "Colbertv2: Effective and efficient retrieval via lightweight late interaction," in *North American Chapter of the Association for Computational Linguistics*, 2021.

[50] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *CoRR*, vol. abs/1807.03748, 2018. arXiv: 1807.03748. [Online]. Available: http://arxiv.org/abs/1807.03748.