修士論文

言語モデルの注意単語を転移する知識蒸留

武田 遥暉

主指導教員 白井 清昭

北陸先端科学技術大学院大学
先端科学技術研究科
（情報科学）

令和 8 年 3 月

Abstract

While current Large Language Models (LLMs) achieve high performance, their enormous number of parameters makes them difficult to deploy in environments with limited computational resources. For this reason, model compression techniques known as knowledge distillation have attracted increasing attention. Knowledge distillation is a technique in which a large model (Teacher model) and an untrained smaller model (Student model) are prepared, and the student model is trained to mimic the internal states of the teacher model on training data samples, thereby reducing the model size while maintaining the performance of the teacher model.

Besides, explaining or visualizing the inference process of LLMs is also important. For example, indicating which words the model attends to during inference can be regarded as a way of explaining the model's reasoning process. However, previous studies have pointed out that the student model does not necessarily attend to the same words as the teacher model during inference. This suggests that the explanatory capability of the student model may be degraded compared to that of the teacher model. In domains such as finance and healthcare, where not only prediction accuracy but also the presentation of the model's reasoning grounds is crucial, a reduction in the explanatory capability of knowledge-distilled models can pose a serious problem.

This study proposes a knowledge distillation method that transfers not only performance on downstream tasks but also the characteristics of the teacher model, such as which attention words it focuses on during inference. Our method extends the existing knowledge distillation method by incorporating a new loss. This new loss serves to minimize the difference between the important words to which the teacher and student models pay attention. It enables us to train a student model that inherits both the inference performance and the explanatory capability of the teacher model.

First, several analyses were performed to examine the differences in attention words between the teacher model and the student model. Using the natural language processing benchmark dataset GLUE, BERT was employed as the teacher model and student models on eight tasks in GLUE were trained using TinyBERT, an existing knowledge distillation method. For both the teacher and student models, the importance of each input word was computed using Integrated Gradients (IG), and words with high values were extracted as the model's attention words. The results of the analyses showed that, even for the same input, the student model tends to perform the tasks by attending to different words than the teacher model. Furthermore, compared with fine-tuned small models obtained by reducing the number of encoder layers in BERT, namely BERT-Medium, BERT-Mini, and

BERT-Tiny, the student models trained with TinyBERT achieved higher task performance, while exhibiting a lower agreement in attention words with the teacher model.

Based on these observations, this study proposes a new knowledge distillation method that aims to increase the agreement of attention words between the teacher model and the student model. Specifically, when training the student model, a new loss function is defined and used such that the student model pays high attention to the words attended to by the teacher model. First, in the same way as the analysis described earlier, the importance of each word is computed using Integrated Gradients. Next, the softmax function is applied to normalize the word importance scores so that their sum over all the input words equals one, obtaining a probability distribution of importance (hereafter referred to as the "importance distribution"). During training of the student model, the difference between the importance distributions of the teacher and student models is minimized. In this study, the similarity between the importance distributions of the two models is measured using the soft Jaccard coefficient, and a new distillation loss is defined as the value obtained by subtracting the soft Jaccard coefficient from one. Then, a new loss function is defined as a weighted sum of this loss and the distillation loss used in the existing TinyBERT method. By updating the model parameters to minimize this loss function, the student model is trained to mimic not only the internal states of the teacher model but also the attention words during inference.

Experiments are carried out to evalute the effectiveness of the proposed method. The teacher model is BERT with 12 Transformer layers, and the student model is either TinyBERT-6L (6 layers) or TinyBERT-4L (4 layers), both of which are knowledge-distilled models obtained during pre-training. The student models are fine-tuned on eight tasks from GLUE. TinyBERT is used as the baseline knowledge distillation method, and the proposed method is compared against it. As evaluation metrics, in addition to task performance metrics such as accuracy, we employ the Jaccard coefficient and a Ranking metric to measure the degree of agreement in attention words with the teacher model. The Jaccard coefficient is defined as the Jaccard coefficient between two sets of top-K important words of the teacher and student models, which are extracted using IG. In contrast, the Ranking metric indicates whether the top-K important words completely match between the teacher and student models, including their ranking (1 if they match, and 0 otherwise). The average of the Jaccard coefficient and Ranking metric over test data samples are computed as evaluation criteria. In addition, the value of K is changed from 1 to 10 and the metrics for each value of K are calculated.

The experimental results show that there is no significant difference in downstream task performance between TinyBERT and the proposed method. This in-

dicates that incorporating the proposed loss does not substantially degrade model performance. For the Jaccard coefficient, improvements were observed across all tasks and all values of K. The proposed method was particularly effective on the SST-2 task (sentiment analysis), where improvements of approximately 20 to 24 points were observed for TinyBERT-6L depending on the value of K, and improvements of approximately 12 to 19 points were observed for TinyBERT-4L. For the CoLA task (linguistic acceptability judgment), the baseline already achieved a high Jaccard coefficient, exceeding 0.86 at Top-9, the proposed method further improved this score by approximately three points. For the Ranking metric, the proposed method performed slightly worse than the baseline for some combinations of tasks and values of K; however, overall, it outperformed the baseline. Since the Ranking metric takes into account the ordering of importance scores, it is a severer evaluation metric than the Jaccard coefficient. Consequently, although the overall values of the Ranking metric were lower, improvements of approximately 10 points were observed in some cases when K ranged from 1 to 3. These results demonstrate the effectiveness of the proposed method in enabling the student model to inherit the ability to attend to similar words in a similar order as the teacher model.

概要

現在の大規模言語モデル (Language Models; LLM) は、高い性能を有する一方で、モデルのパラメタ数が膨大であるため、計算資源の限られた環境においては利用が困難である。このため知識蒸留と呼ばれるモデル圧縮手法が注目されている。知識蒸留は、大規模な言語モデル (教師モデル) と未学習の小規模モデル (生徒モデル) を用意し、訓練データのサンプルに対する教師モデルの内部状態を模倣することで、教師モデルの性能を維持しつつモデルのサイズを小さくする手法である。

一方、LLM による推論の過程を説明あるいは可視化することも重要である。例えば、推論時にモデルが注意している単語を示すことは、モデルによる推論の過程を説明するものとみなせる。しかしながら、生徒モデルが教師モデルと同じ単語に注意して推論を行っているとは限らないことが先行研究により指摘されている。これは生徒モデルの説明能力が教師モデルと比べて低下している可能性があることを意味する。金融分野や医療分野など、精度だけではなくモデルの推論根拠の提示も重要な分野においては、知識蒸留モデルの説明能力の低下は問題となりうる。

本研究は、下流タスクに対する性能を維持するだけでなく、推論時にどのような単語に注意するかといった教師モデルの特性も転移する知識蒸留の手法を提案する。既存の知識蒸留の手法を拡張し、教師モデルと生徒モデルが注意する単語を定量化し、これを新しい損失項として組み込むすることで、教師モデルの推論性能と説明能力の両方を継承する生徒モデルを学習することを狙う。

まず、教師モデルと生徒モデルの注意単語の違いを検証する実験を行った。自然言語処理のベンチマークデータセット GLUE を用い、BERT を教師モデルとし、8 つのタスクに対して既存知識蒸留手法である TinyBERT を用いて生徒モデルを学習した。教師モデルと生徒モデルのそれぞれについて、Integrated Gradients (IG) を用いて入力における各単語の重要度を計算し、これが大きい単語をモデルの注意単語として抽出した。検証の結果、同じ入力に対しても生徒モデルと教師モデルとで異なる単語に注意して推論を行っている傾向が強いことが確認された。また、BERT の Encoder レイヤー数を削減した小規模モデルである BERT-Medium、BERT-Mini、BERT-Tiny をファインチューニングしたモデルと比べて、TinyBERT による生徒モデルはタスクを解く能力が高い一方で、教師モデルとの注意単語の一致率は低くなることが確認された。

これを踏まえ、本研究では、教師モデルと生徒モデルの注意単語の一致率を高めるための新たな知識蒸留手法を提案する。具体的には、生徒モデルを学習する際、教師モデルが注意する単語に対して生徒モデルも高い注意を払うよう損失関数を設計する。まず、先に述べた分析と同じように、IG を用いて単語の重要度を計算する。次に、Softmax 関数を用いて入力における単語の重要度スコアの和が 1 になるように正規化し、単語の重要度の確率分布 (以下、「重要度分布」と呼ぶ) を得る。生徒モデルを学習する際には教師モデルと生徒モデルの重要度分布の差

を最小化する。本研究では Soft Jaccard 係数を用いて両モデルの重要度分布の類似度を測り、1 から Soft Jaccard 係数を引いた値を新しい蒸留損失項として定義する。この損失項と既存手法の TinyBERT における蒸留損失項の重み付き和を新しい損失関数として定義する。この損失関数を最小化するようにパラメタを更新することで、教師モデルの内部状態だけではなく推論時の注意単語も模倣するよう生徒モデルが学習される。

提案手法の評価実験を行う。教師モデルは BERT(Transformer 層が 12 層)、生徒モデルは事前学習時に知識蒸留されたモデルである TinyBERT-6L(6 層) または TinyBERT-4L(4 層) とし、GLUE の 8 つのタスクを対象に生徒モデルをファインチューングする。ベースラインの知識蒸留手法を TinyBERT とし、提案手法と比較する。評価指標として、タスクの性能の評価指標 (正解率など) に加え、注意単語の教師モデルとの一致度を測るために Jaccard 係数と Ranking 指標を用いる。Jaccard 係数は IG によって上位 K 個の重要単語を抽出したとき、その重要単語集合の Jaccard 係数である。一方、Ranking 指標は上位 K 個の重要単語が順位も含めて完全に一致している (1) か否か (0) を表す。モデルの評価時にはテストデータのサンプルに対する Jaccard 係数ならびに Ranking 指標の平均値を算出する。また、K の値を 1 から 10 まで変動させ、それぞれの値ごとに指標を算出する。

実験の結果、TinyBERT と提案手法とで下流タスクに対するモデルの性能に大きな差はなかった。これは本研究で提案する損失項を組み込んでもモデルの性能が大きく損われることがないことを意味する。Jaccard 係数については全てのタスクと K の値において改善が見られた。特に有効だったのは SST-2 タスク (感情分析タスク) であり、TinyBERT-6L については K の値によって 20〜24 ポイント、TinyBERT-4L については 12〜19 ポイント程度の改善が見られた。CoLA タスク (文の妥当性判定タスク) ではベースラインでも Jaccard 係数が高く、Top-9 では 0.86 以上であったが、提案手法では更に 3 ポイント向上した。Ranking 指標については、一部のタスクと K の組について提案手法がベースラインと比べて低かったが、全体的にはベースラインを上回った。Rankinig 指標は重要度スコアの順序も考慮するため、Jaccard 係数と比べて厳しい評価指標である。したがって全体的に Rankinig 指標の値は低かったものの、K が 1〜3 の場合については 10 ポイント程度の改善が見られるケースもあった。以上から、同じような単語を同じような順序で注意する能力を教師モデルから継承するという点での提案手法の有効性が確認された。

目 次

図 目 次

表 目 次

第1章　はじめに

1.1　背景

大規模言語モデル（Large Language Model; LLM）は、大規模テキストで事前学習された言語モデルであり、様々な言語タスクにおいて高い性能を示している。特に文書分類や感情分析などの分類モデルとして用いる場合、入力文全体の意味情報を統合した表現を利用することで、既存手法を上回る成果を挙げている。

このような課題に対し知識蒸留と呼ばれる手法が盛んに研究されている。知識蒸留とは、LLM のようなパラメタ数の多い高性能なモデルを教師モデル、より少ないパラメータを持つモデルを生徒モデルとし、教師モデルの優れたタスク遂行能力を生徒モデルへ転移する手法である。特に分類タスクにおいては、性能をほぼ維持したままモデルサイズを半分程度まで削減可能であるなど、高い有効性が示されている。現在、知識蒸留はモデル軽量化の代表的な手法として位置付けられている。

しかしながら、知識蒸留は、計算資源の制約下において教師モデルの代替として利用可能な生徒モデルを構築することを目的とした技術であるにもかかわらず、知識蒸留によって学習された生徒モデルは分類精度を維持できている一方、推論時に重要と判断する単語が教師モデルとで一致しないことが報告されている [6]。これは生徒モデルが教師モデルとは異なる特徴に着目してタスクを解いていることを意味する。一方、LLM に単に入力を分類させるだけでなく、なぜそのような分類結果になったのかを説明させることも重要である。しかし、生徒モデルが教師モデルと異なる特徴に着目していることは、生徒モデルの説明可能性が低下していることを示唆する。このことは、医療分野や金融分野など判断根拠の信頼性が重視される領域において致命的な問題となり得るため、生徒モデルを教師モデルの単純な代替として用いることには依然として課題が残っている。

1.2　目的

本研究では、教師モデルの分類性能を維持するだけでなく、教師モデルが分類時に注意する単語を維持しながら知識を生徒モデルに転移する知識蒸留の手法を開発することを目的とする。分類タスクを対象とし、以下の手順にしたがって知識蒸留を実現する。

まず、目的のタスクに対し十分に学習された教師モデルと、教師モデルよりも小規模な未学習の生徒モデルを準備し、両モデルに同一の入力インスタンスを与えて推論 (分類タスクの実行) を行う。次に、得られた推論結果に対して Integrated Gradients を用いて単語重要度を算出し、それらを重要度分布として表現する。最後に、教師モデルと生徒モデルの重要度分布が近づくよう損失関数を設計し、先行研究で提案されている知識蒸留手法に本研究で提案する損失関数を導入することで、生徒モデルを学習する。本手法により、る損失関数を導入することで、生徒モデルを学習する。本手法により、分類性能を先行研究と同程度に維持したまま、推論時に重要と判断される単語に関して教師モデルとの整合性を向上させることを狙う。

1.3 本論文の構成

本論文の構成は以下の通りである。2 章では先行研究について述べる。3 章では知識蒸留によって得られた生徒モデルが教師モデルとどの程度同じ単語に注意しているかを分析する。4 章では 3 章の分析結果を踏まえ、生徒モデルが教師モデルと同じ単語に注意して推論を行うように生徒モデルを学習する知識蒸留手法を提案する。5 章では、本研究の手法についての評価について述べる。最後に 6 章で本研究のまとめと今後の課題について述べる。

第2章　関連研究

本章では本研究の関連研究について述べる。2.1 節では知識蒸留の基礎について説明する。2.2 節では事前学習済み言語モデルについて説明する。2.3 節では知識蒸留によって構築されたモデルについて説明する。2.4 節ではモデルの推論根拠の解釈に関する研究について説明する。2.5 節では知識蒸留モデルと小規模モデルの推論根拠の違いに関する研究について説明する。最後に 2.6 節では本研究の特徴について説明する。

2.1　知識蒸留

本節で述べる研究トピックは知識蒸留 (Knowledge Distillation)[2] である。知識蒸留は、性能が高くパラメータ数も多いモデルを教師モデルとし、パラメータ数が少ないモデルへとタスクを解くための能力を転移する技術である。これまでに様々な知識蒸留手法が提案されてきており、教師モデルの出力層の知識を転移するもの、隠れ状態から知識を転移させるもの、出力層と隠れ層の両方から知識を転移させるものなど、様々な知識蒸留のアプローチが提案されてきた。

ここでは、分類タスクを例に、知識蒸留の基本的な手法を説明する。分類タスクでは、通常、数百から数万のラベル付きインスタンスが与えられる。代表的な自然言語理解ベンチマークである GLUE[12] は、複数の分類タスクから構成されるデータセット群であり、知識蒸留の評価においても標準的に用いられる。例えば、GLUE に含まれる MRPC（Microsoft Research Paraphrase Corpus）は、2 つの英文が意味的に等価であるかどうかを判定する二値分類タスクである。各インスタンスにはラベル $y \in \{0, 1\}$ が付与されており、意味的に等価な場合を 1、そうでない場合を 0 と定義する。MRPC におけるサンプルの例を以下に示す。

> 例
> sentence1: Amrozi accused his brother, whom he called “the witness”, of deliberately distorting his evidence.
> sentence2: Referring to him as only “the witness”, Amrozi accused his brother of deliberately distorting his evidence.
> label: equivalent (1)

通常の分類モデルの学習では、生徒モデル f_s が入力 x に対して予測するクラス確率分布 $p_s(y \mid x)$ と、正解ラベル y との間のクロスエントロピー損失を最小化

する。この損失関数は次式で定義される。

$$\mathcal{L}_{\mathrm{CE}} = -\sum_{c} y_c \log p_s(c \mid x) \tag{2.1}$$

ここで、c はクラスを表し、y_c は正解クラスに対応する数値 (1 または 0) である。

知識蒸留では、この正解ラベルに基づく学習に加えて、同一の入力 x に対する教師モデル f_t の出力分布 $p_t(y \mid x)$ を生徒モデルが模倣することを目的とする。教師モデルと生徒モデルの出力分布の差を測る指標として Kullback–Leibler（KL）ダイバージェンスが一般に用いられる。このとき、知識蒸留損失は、次式で表される。

$$\mathcal{L}_{\mathrm{KD}} = \mathrm{KL}\big(p_t(\cdot \mid x) \,\|\, p_s(\cdot \mid x)\big) \tag{2.2}$$

ここで、KL は 2 つの確率分布の KL ダイバージェンスを、$\cdot$ はクラス全体を表し、教師モデルと生徒モデルの確率分布が近くなるように損失が設計されていることを意味する。

最終的な学習では、正解ラベルに基づくクロスエントロピー損失と、教師モデルの振る舞いを模倣する知識蒸留損失を組み合わせ、以下の損失関数を最小化する。

$$\mathcal{L} = (1-\lambda)\mathcal{L}_{\mathrm{CE}} + \lambda \mathcal{L}_{\mathrm{KD}} \tag{2.3}$$

ここで、$\lambda \in [0, 1]$ はハイパーパラメータであり、2 つの損失のうち知識蒸留損失をどれだけ重視するかを制御する。

このように知識蒸留では、データセットに明示的に含まれるラベル情報だけでなく、教師モデルが獲得した予測分布という暗黙的な知識を生徒モデルに転移することで、高い性能を維持したままモデルの軽量化を実現している。

2.2 事前学習済み言語モデル

分類モデルの知識蒸留における教師モデルとしては BERT を用いられることが多い。本節では BERT の詳細について述べる。

2.2.1 BERT

BERT[1] は、Transformer[11] を基盤とした双方向言語モデルであり、大規模コーパスを用いた事前学習によって汎用的な言語表現を獲得する。BERT の事前学習では、Masked Language Model（MLM）と Next Sentence Prediction（NSP）の 2 つのタスクが用いられる。

MLM では、入力文中の一部の単語をマスクし、その単語を周辺文脈から予測することで単語表現を学習する。MLM の損失関数は以下のように定義される。

$$\mathcal{L}_{\mathrm{MLM}} = -\sum_{i=1}^{N} \log p(t_i \mid T_i) \tag{2.4}$$

ここで、t_i はマスクされた単語、T_i は t_i 以外の単語からなる入力文を表す。

次に、NSP（Next Sentence Prediction）は、2 つの文が元の文書中で連続しているか否かを判定する二値分類タスクであり、文間の関係性を学習することを目的としている。事前学習時には、文書中で実際に連続する文の組を正例（$y = 1$）、無関係な文の組を負例（$y = 0$）として与える。入力文の組の全体を表す `[CLS]` トークンの最終層表現を用いて、2 文が連続している確率 P_{NSP} を推定する。NSP の損失関数は、二値交差エントロピー損失として次式で定義される。

$$\mathcal{L}_{\mathrm{NSP}} = -\left[y \log P_{\mathrm{NSP}} + (1 - y) \log(1 - P_{\mathrm{NSP}})\right] \tag{2.5}$$

ここで、$y \in \{0, 1\}$ は 2 文が連続しているか否かを示す正解ラベルであり、P_{NSP} はモデルによって推定された連続確率 (2 つの文が連続して出現する確率) を表す。このように NSP を導入することで、BERT は単語レベルの意味情報だけでなく、文脈の整合性を捉えた表現を獲得することが可能となる。

BERT の事前学習後、下流タスクに対するファインチューニングが行われる。入力文の先頭に特殊トークン [CLS] を置き、このトークンに対応する BERT の埋め込みを入力全体の抽象表現とみなす。これを全結合層に渡し、下流タスクにおける分類クラスの出力を得る。モデル全体を下流タスクのラベル付きデータを用いて再学習することで、下流タスクに対する分類性能を向上させる。

2.2.2 本研究における教師モデルとしての BERT

本研究では教師モデルとして BERT を用いる。先に述べたように、事前学習された BERT を下流タスクに適用させるためにはファインチューニングが行われる。GLUE 内における 8 つの分類タスクに対してファインチューニングを行い、各タスクに対して十分に学習された BERT モデルを教師モデルとして用いる。

2.3 知識蒸留によって構築されたモデル

既に述べたように、知識蒸留は、大規模な教師モデルが出力する確率分布を小規模な生徒モデルに模倣させることで、教師モデルに近い性能を持つ小規模モデルを獲得することを目的としている。当初は、全結合層を最終層に持つディープニューラルネットワークを対象に、教師モデルの出力層で得られるロジットやクラス確率を用いて生徒モデルを学習させる手法が中心であった。その後、Transformer ベースの言語モデルが台頭すると、どの層の情報をどのように移すのかを工夫した蒸留法が多数検討され、中間層・Attention 重みまで含めて知識を転移する研究が進展した。

DistilBERT[8] は BERT を教師モデルとした代表的な知識蒸留モデルである。DistilBERT は、ファインチューニングの段階における知識蒸留の適用だけでなく、

事前学習においても、事前学習済みの BERT を教師モデルとして、その内部表現や出力を模倣する生徒モデルとして訓練されている。BERT のパラメータ数を約 40%削減しながらも、多くの下流タスクにおいて高い性能を維持している。

TinyBERT[3] は、BERT を教師モデルとした知識蒸留によって構築された軽量な言語モデルである。TinyBERT は、教師モデルの出力層だけでなく、中間層の隠れ状態や Attention 機構の重みも模倣することで、モデルサイズを大幅に削減しながら高い性能を維持している。TinyBERT の蒸留は大きく 2 段階で行われる。

事前学習段階 (pre-training) 事前学習段階では、BERT の事前学習で用いられた Masked Language Model（MLM）や Next Sentence Prediction（NSP）に加えて、中間層の隠れ状態や Attention 重みの模倣を目的とした損失を導入する。これにより、生徒モデルは教師の内部表現を効率的に学習し、一般的な言語の抽象表現を獲得する。

下流タスクのファインチューニング 下流タスクへの適応はさらに 2 段階（段階的蒸留）で実施される。

- **第 1 段階 — 中間層蒸留 (intermediate layer distillation)** まず中間層の隠れ状態や Attention 重みの模倣損失を導入し、生徒の内部表現を教師に近づけることで下流タスクへ適応させる。
- **第 2 段階 — 出力層蒸留 (prediction layer distillation)** 続いて教師の出力（ソフトラベル）とデータのハードラベルを併用して生徒の出力層を学習させ、教師モデルの推論能力を模倣させる。

さらに、TinyBERT の学習では、下流タスクでの性能を高めるためにデータ拡張を併用することが知られている。具体的には、元のデータセットに対してランダムに単語を挿入・削除・置換することで多様な入力インスタンスを生成し、頑健性を向上させる。

上記の手続きで学習された TinyBERT は、BERT の知識を効率的に圧縮し、軽量ながら高性能なモデルとなっている。

2.4 モデルの推論根拠の解釈に関する研究

近年、深層学習モデルは画像認識や自然言語処理などの多くのタスクにおいて人間を凌駕する性能を達成している。しかし、その高い性能の代償として、モデル内部の計算プロセスは複雑化し、人間にとって理解困難な「ブラックボックス」となっていることで、モデルの推論における根拠となる特徴量を明示的に把握することが難しいという課題が存在する。

この要求に応えるための技術として、特徴量帰属法（Feature Attribution）が研究されている。特徴量帰属法は、モデルの予測結果に対する入力特徴量（例えば、画像の各ピクセルやテキストの各単語）の寄与度を定量的に算出する手法の総称である。寄与度が高い特徴量は、モデルが予測を行う上で重要な手がかりとして利用したことを示唆する。

本節では、まず代表的な特徴量帰属法である Integrated Gradients[9] を紹介する。Integrated Gradients は、ある入力 x と、情報を持たない参照点であるベースライン x' との間の経路に沿って勾配を積分することで定義される。入力ベクトル $x \in \mathbb{R}^n$ における i 番目の特徴量 x_i の寄与度 $\mathrm{IG}_i(x)$ は以下の式で計算される。

$$\mathrm{IG}_i(x) = (x_i - x_i') \int_{\alpha=0}^{1} \frac{\partial F(x' + \alpha(x - x'))}{\partial x_i} \, d\alpha \tag{2.6}$$

ここで、F は微分可能な機械学習モデル（例えばニューラルネットワークのソフトマックス出力値）、x' はベースラインベクトルを表す。α は補間係数であり、$[0, 1]$ の範囲で変化することで、ベースライン x' から入力 x への直線経路を辿る。この経路積分により、IG は以下の重要な公理的性質を満たす。

感度 (Sensitivity)　ある特徴量のみを変化させたときにモデルの出力が変化する場合、その特徴量は非ゼロの寄与度を持つべきであるという公理である。従来の勾配法では、ReLU や Sigmoid などの活性化関数が飽和領域（勾配がほぼ 0 となる領域）に入ると、入力が変化しても勾配が消失し、寄与度が 0 と計算されてしまう問題があった。Integrated Gradients は、ベースライン入力から実入力までの経路に沿って勾配を積分することで、経路上のどこかで勾配が存在すればその影響を累積的に捉えることが可能となり、単純勾配法では満たされない感度の公理を満たす。

完全性 (Completeness)　完全性とは、Integrated Gradients（IG）によって算出された各入力次元の寄与度の総和が、入力 x に対するモデルの出力値とベースライン x' に対するモデルの出力値との差分に一致するという性質である。すなわち、次式が成り立つ。

$$\sum_{i=1}^{n} IG_i(x) = F(x) - F(x') \tag{2.7}$$

この性質は、モデルの予測値がどの程度変化したかという情報を、算出された寄与度の合計によって過不足なく説明できることを意味している。そのため Integrated Gradients による説明がモデルの挙動と整合的であることを保証する重要な公理の一つである。

Integrated Gradients は、入力が連続的に変化し、かつ微分可能であることを前提とした手法である。しかし、自然言語処理においてモデルへの入力は離散的

なトークン列であり、入力そのものを連続的に変化させることはできない。この問題を回避するため、自然言語処理タスクでは、単語を連続値ベクトルに変換する埋め込み層の出力を入力とみなし、その埋め込みベクトルに対して Integrated Gradients を適用する方法が一般的に用いられる。

ある単語 w_t の重要度を評価するためには、まずその単語に対応する埋め込みベクトルの各次元 j に対して Integrated Gradients 値を計算する。埋め込み次元 j における IG 値は、以下の式で定義される。

$$IG_{t,j} = (e_{t,j} - b_{t,j}) \int_0^1 \frac{\partial F(B + \alpha(E - B))}{\partial e_{t,j}} \, d\alpha \tag{2.8}$$

ここで、E は入力文全体の埋め込み行列、B はベースラインに対応する埋め込み行列を表す。この式は、ベースラインの埋め込みから入力の埋め込みへと徐々に特徴量を変化させながら、その過程におけるモデル出力の変化に対する勾配を積分することで、各埋め込み次元の寄与度を評価していることを意味している。

最終的に、単語 w_t 全体としての重要度スコア S_t を得るためには、埋め込みベクトルの各次元に対して算出された IG 値を集約する必要がある。こうして得られたスコア S_t が大きい単語ほど、モデルの予測結果に強い影響を与えた重要な単語であると解釈できる。

Integrated Gradients の計算において、一般的には、積分区間 $[0, 1]$ を m 個のステップに分割し、リーマン和によって近似を行う。このとき、Integrated Gradients は次式のように近似される。

$$IG_i(x) \approx (x_i - x'_i) \frac{1}{m} \sum_{k=1}^{m} \frac{\partial F\left(x' + \frac{k}{m}(x - x')\right)}{\partial x_i} \tag{2.9}$$

ここで、m は積分の近似に用いるステップ数を表す。各ステップごとに勾配計算を行う必要があるため、Integrated Gradients の計算コストはステップ数 m に比例して増加する。ステップ数が小さい場合には積分近似誤差が大きくなる一方で、ステップ数を増やすことで精度は向上するが計算時間も増加する。そのため、実用上は計算コストと精度のトレードオフを考慮し、適切なステップ数を設定する必要がある。

2.5 知識蒸留モデルと小型モデルの推論根拠の違いに関する研究

近年、知識蒸留モデルと小型モデルの推論根拠の違いに関する研究が進んでいる。Rai らは、知識蒸留によって学習された小型モデルが、教師モデルと異なる単語に注目して推論を行うことを報告している [6]。この研究では、BERT を教師モデルとして蒸留した DistilBERT と、BERT-Medium、BERT-Mini、BERT-Tiny

といったBERTを小型化したモデルとを比較し、LIME[7]やSHAP[4]といったモデル解釈説明可能性手法を用いて、各モデルが推論において重要視する単語を分析した。その結果、精度に関しては知識蒸留モデルが小型モデルよりも高い性能を示す一方で、推論根拠に関しては小型モデルの方が教師モデルと高い一致度を示すことを明らかにした。この研究は、知識蒸留モデルが教師モデルの単語重要度分布を十分に模倣できていないことを示しており、知識蒸留における推論根拠の転移という課題を提起している。すなわち、知識蒸留モデルが教師モデルと同程度の精度を示したとしても、意思決定の根拠となる特徴(入力中の重要語)が異なる場合、予測の妥当性や説明の一貫性が担保されない可能性がある。

2.6 本研究の特徴

従来の知識蒸留手法は、教師と生徒のモデル出力(ロジット)の分布を一致させることを目的としてきた。これに対し、本研究は個々の単語がモデルの最終的な予測判断にどの程度貢献したのかを示すトークン重要度分布に焦点を当てる。Integrated Gradientsを用いてトークン重要度を定量化し、教師モデルと生徒モデルが同じ単語に注目して予測を行うよう学習させることで、「モデルの推論根拠」を明示的に転移させ、精度だけでなく信頼性と解釈可能性を備えた知識蒸留を実現する。本研究は知識蒸留における教師モデルと生徒モデルの重要単語の違いに着目したRaiらの研究[6]に着想を得ているが、単語の重要度としてLIMEやSHAPの代わりにIntegratedGradientsを用いている点、またモデルが重要視する単語の教師モデルから生徒モデルへの転移を試みている点が異なる。

提案手法により、分類の際に注意するトークンの重要度についても教師モデルの優れた性能を生徒モデルに継承させる。教師モデルと生徒モデルのトークン重要度分布を合わせることで、生徒モデルの推論プロセスが高度化され、推論プロセスが重視される場面での知識蒸留モデルの信頼性を向上させる。

第3章　言語モデルの知識蒸留に関する分析

本章では知識蒸留に関する実験的な分析を行う。先に述べたように、従来の知識蒸留では、教師モデルの性能を維持することを目的としており、教師モデルと生徒モデルとで推論の際にモデルが重要であると認識する単語には違いがあることが報告されている。まず、言語モデルとして TinyBERT を選び、いくつかの自然言語処理タスクを対象に知識蒸留を行い、その結果を報告する。次に、TinyBERT とその教師モデルである BERT との注意単語の違いについて分析を行う。

3.1　タスク

本実験では、自然言語処理評価用データセット GLUE における 8 つの分類タスクを対象とする。8 つのタスクの名称とタスクの定義を以下の通りである。

CoLA 文の文法的妥当性判定。与えられた文が文法的に自然（acceptable）か否かを判定する二値分類タスク

SST-2 文の感情判定（感情分析）。短いレビュー文が肯定的か否定的かを判定する二値分類タスク

MRPC パラフレーズ判定。2 つの文が意味的に同義（paraphrase）かどうかを判定する二値分類タスク

STS-B 文対の意味的類似度評価。2 文の意味的類似度を連続値（0–5）で評価する回帰タスク

QQP 質問ペアの重複判定。ウェブ上の QA サイト Quora に投稿された 2 つの質問が同じ意図・内容かを判定する二値分類タスク

MNLI マルチジャンルの自然言語推論 (Multi-Genre Natural Language Inference)。前提と仮説の関係が含意（entailment）、矛盾（contradiction）、中立（neutral）のいずれかを判定する三値分類タスク

QNLI 質問と文の関連判定。与えられた文が質問に対する答えや関連情報を含むかどうかを判定する二値分類タスク

RTE 含意関係認識タスク。前提と仮説の含意関係（含意／非含意）を判定する二値分類タスク

各タスクのインスタンス例 (入力と正解ラベル) を表 3.1 に示す。

表 3.1: GLUE における各タスクのサンプルとラベル (STS-B では文間類似度) の例

タスク	入力例	ラベル
CoLA	“He go to the store.”	unacceptable
SST-2	“The movie was fantastic and deeply moving.”	positive
MRPC	“The company reported profits.” “The firm announced earnings.”	equivalent
STS-B	“A man is playing a guitar.” “Someone plays a stringed instrument.”	4.5
QQP	“How to bake a cake?” “What is the process to make a cake?”	duplicate
MNLI	前提: “A dog is sleeping on the couch.” 仮説: “An animal is resting on furniture.”	entailment
QNLI	質問: “Who wrote Hamlet?” 文: “Hamlet was written by Shakespeare.”	relevant / entailment
RTE	前提: “The sky is clear and blue.” 仮説: “The sky has no color.”	not-entailed

3.2 知識蒸留の効果の検証

本節では、GLUE の各タスクにおいて、おいて、TinyBERT で知識蒸留したモデルの性能を評価する。同様の検証は既に行われているが、自身の計算機環境で改めて検証する。TinyBERT は、BERT の事前学習モデルから蒸留された事前学習モデルが必要である。この事前学習済み蒸留モデルを初期モデルとして、GLUE の各タスクに対して再度蒸留を行うことで各タスクに適応させる必要がある。本来であれば、この事前学習の時点から BERT モデルを蒸留する必要があるが、本実験では事前に公開されている TinyBERT の汎用事前学習モデルを利用することで、この手続きを省略する。本実験では TinyBERT の公開事前学習モデルとして TinyBERT-4L と TinyBERT-6L を用いる。これらはモデル内部ののエンコーダ層の数がそれぞれ 4 層、6 層であることを示す。教師モデルとして、BERT-base を GLUE の各タスクに対してファインチューニングしたモデルを用いる。

表 3.2 に教師モデルの性能と蒸留した TinyBERT-4L と TinyBERT-6L の性能を示す。評価指標は、回帰タスクである STS-B タスクは Pearson correlation、分類

タスクである CoLA は Matthews correlation、それ以外の分類タスクは accuracy である。これらは GLUE で採用されている評価指標である。なお、MNLI タスクは MNLI-m と MNLI-mm の 2 つのテストセットがあるため、これら 2 つの実験結果を載せている。MNLI-m は訓練データと同じドメインのサンプルから構成されるテストセットであり、MNLI-mm は異なるドメインのサンプルから構成されるテストセットである。それぞれ in-domain(訓練データとテストデータのドメインが同じ) と out-of-domain(訓練データとテストデータのドメインが異なる) の設定でのモデルの評価に用いる。

表 3.2: 教師モデル (BERT-base) と TinyBERT の性能比較

Model	**MRPC**	**CoLA**	**STS-B**	**RTE**	**QQP**	**MNLI-m**	**MNLI-mm**	**QNLI**	**SST-2**
Metric	accuracy	matthews	pearson	accuracy	accuracy	accuracy	accuracy	accuracy	accuracy
BERT-base(FineTuned)	88.245	53.393	87.952	72.922	90.883	84.554	84.469	91.453	92.433
TinyBERT-6L	86.275	48.724	87.701	63.899	90.873	82.425	82.669	90.335	91.743
TinyBERT-4L	85.539	37.480	87.407	63.177	90.124	81.803	82.170	87.058	90.940

TinyBERT-6L、TinyBERT-4L ともに、CoLA タスクにおいて、知識蒸留モデルの顕著な性能低下が見られる。特によりサイズの小さい 4 層のモデルで教師モデルと大きな差がある。一方、それ以外のタスクでは、蒸留モデルは教師モデルと同程度の性能を示すことが確認できた。

3.3 モデルの注意単語の違いの分析

本節では、知識蒸留されたモデル (生徒モデル) とその教師モデルとで、下流タスクを解く際にモデルが注目 (注意) している単語の差異を分析する。また、知識蒸留したモデルと同程度のパラメタ数を持つ小型の言語モデルについても注意単語の違いを分析する。

教師モデルとして、下流タスクに対してファインチューニングした BERT を用いる。知識蒸留したモデルとして TinyBERT-6L と TinyBERT-4L を用いる。これらは 3.2 節で検証したモデルと同じである。小型の言語モデルとして、事前学習済みの BERT-Medium、BERT-Mini、BERT-Tiny を下流タスクでファインチューニングしたモデルを用いる。これらのモデルはぞれぞれ、8 層、4 層、2 層のエンコーダ層を持ち、隠れ次元数もそれぞれ 512、256、128 に削減され、モデルサイズが縮小されている [10]。本分析では、知識蒸留モデルとファインチューニングされた小型モデルとの注意単語の違いも考察する。

まず、GLUE のそれぞれのタスクに対する各モデルの性能を表 3.3 に示す。BERT-base、TinyBERT-6L、TinyBERT-4L の結果は表 3.2 の再掲である。TinyBERT-4L は 4 層しかないにもかかわらず、6 層の BERT-Medium と同等以上の性能を多くのタスクで達成している。これは知識蒸留によって、より少ないパラメータで教師モデルの知識を学習できていることを示す。すなわち、知識蒸留を用いたモデ

ルの方が、単純に層数を減らしたモデルよりも高い性能を発揮できることを示しており、知識蒸留が単なるモデル縮小ではなく効果的な知識の転移を実現していることの証左である。

表 3.3: GLUE の下流タスクに対する教師モデル、知識蒸留モデル、小型モデルの性能の比較

Model	**MRPC**	**CoLA**	**STS-B**	**RTE**	**QQP**	**MNLI-m**	**MNLI-mm**	**QNLI**	**SST-2**
Metric	accuracy	matthews	pearson	accuracy	accuracy	accuracy	accuracy	accuracy	accuracy
BERT-base(FineTuned)	88.245	53.393	87.952	72.922	90.883	84.554	84.469	91.453	92.433
TinyBERT-6L	86.275	48.724	87.701	63.899	90.873	82.425	82.669	90.335	91.743
TinyBERT-4L	85.539	37.480	87.407	63.177	90.124	81.803	82.170	87.058	90.940
BERT-Medium	82.100	47.640	84.126	68.230	90.380	80.640	80.610	82.730	91.730
BERT-Mini	76.470	29.380	82.430	61.730	89.834	79.010	78.520	81.510	89.230
BERT-Tiny	69.850	12.240	79.850	61.010	84.210	69.790	70.400	79.160	79.340

次に、生徒モデルが推論時に教師モデルとどれだけ同じような単語に注意しているかを測るスコアを計算する。スコアの計算方法の詳細は 5.1 節で述べるが、ここではその概略を述べる。まず、テストデータのそれぞれのインスタンスに対して各単語の Integrated Gradients を計算する。2.4 節で述べたように、Integrated Gradients は各単語のモデルの注意度を表す指標である。Integrated Gradients によって算出された重要度スコアの大きい順に入力単語を並べ、それが大きい上位 K 個の単語を抽出する。これをその入力インスタンスに対する重要単語集合と定義する。教師モデルと生徒モデルもしくは小型モデルのそれぞれについて、上記の手続きで重要単語集合を抽出し、それらの類似度を測る。集合間の類似度を測る指標として Jaccard 係数を用いる。Jaccard 係数が大きい生徒モデルほど、教師モデルと同じ単語に注目し、したがって教師モデルと同程度の説明能力を持つとみなす。最終的に、テストデータのインスタンスに対して教師モデルと生徒モデルの重要単語集合の類似度を測り、その平均値を算出する。以下、この指標を「Top-K Jaccard 係数」と呼ぶ。このスコアは、教師モデルが入力のどの単語に注意するかの振舞いを生徒モデルにどの程度転移できたかを示すものである。ただし、知識蒸留を行わない小型モデルについては、Top-K Jaccard 係数は教師モデルと小型モデルの注意単語の近さを表す指標になる。今回の分析では、重要単語の数 K を 1 から 10 まで変化させ、それぞれの K 毎に算出した Top-K Jaccard 係数を評価する。

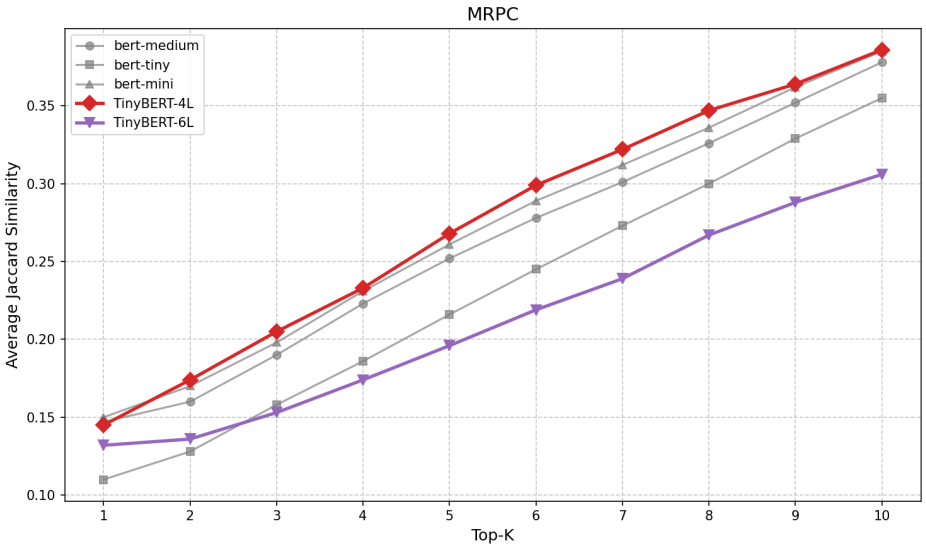
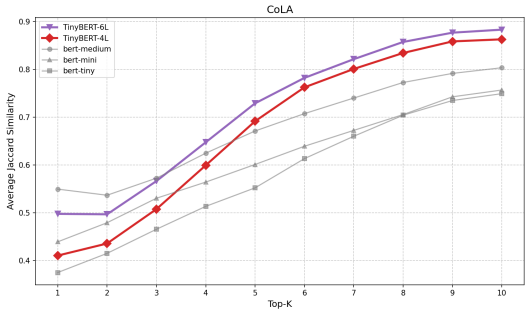
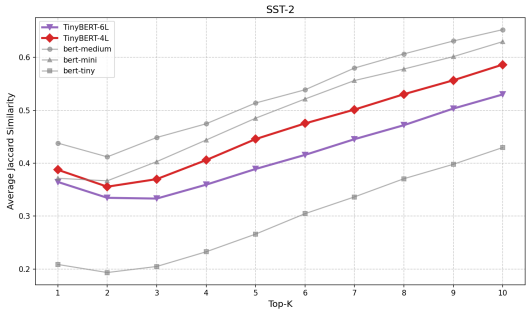


図 3.1: MRPC タスクにおける Top-K Jaccard 係数

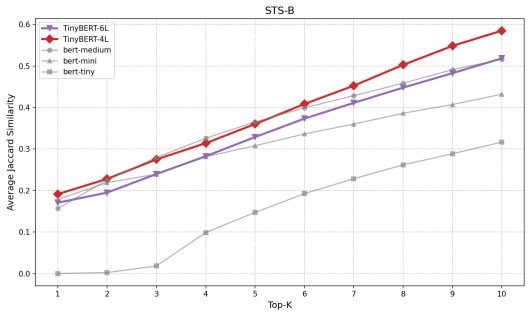
図 3.3 に MRPC タスクにおける知識蒸留モデルならびに小型モデルの Top-K Jaccard 係数を示す。横軸は K を、縦軸は Top-K Jaccard 係数を表す。全体的に Top-K Jaccard 係数が低く、生徒モデルや小型モデルが教師モデルと同じような単語に注目していないことがわかる。また、表 3.3 に示すように、教師モデル以外では TinyBERT-6L の性能が一番高いが、Top-K Jaccard 係数は一番低い。特に一番精度が低い BERT-tiny よりも注意単語の一致度が低いことが確認できる。このことは、知識蒸留によって得られたモデルの下流タスクに対する性能が高くても、入力の中で注目する単語が教師モデルと異なる例となっている。



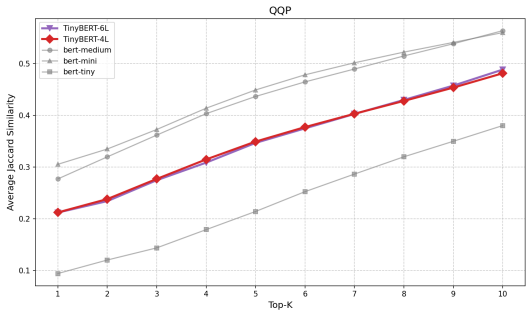
(a) CoLA



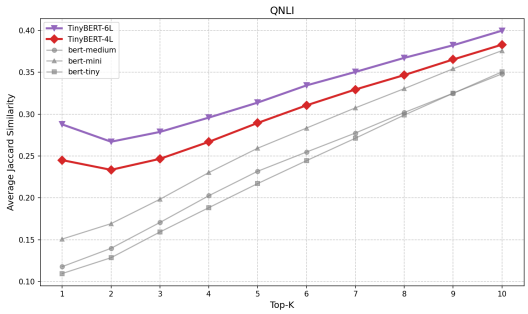
(b) SST-2



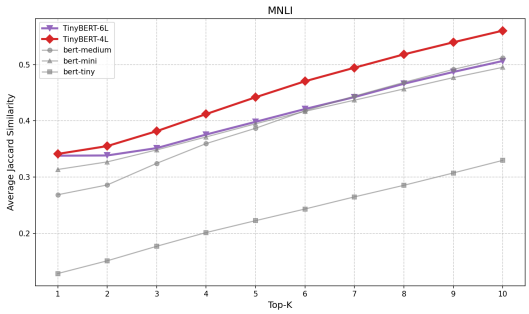
(c) STS-B



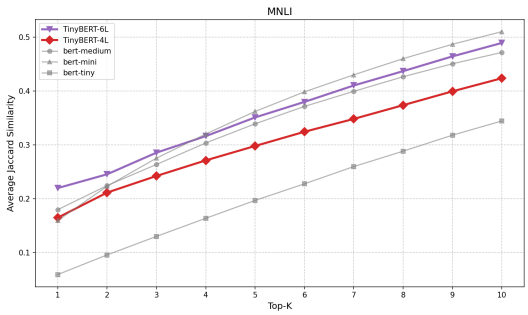
(d) QQP



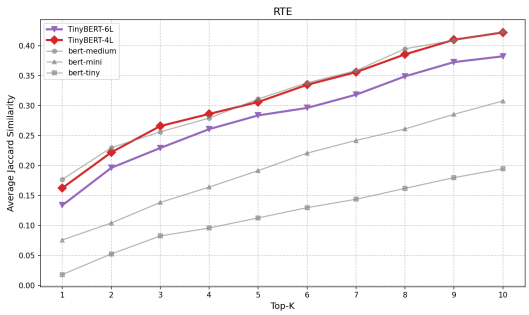
(e) QNLI



(f) MNLI-m



(g) MNLI-mm



(h) RTE

図 3.2: 他タスクにおける注目単語転移スコア

他のタスクに対する実験結果を図 3.2 に示す。QQP、SST-2 に関しては MRPC と同様に知識蒸留モデルが小型モデルよりも Top-K Jaccard 係数が低くなる傾向が

みられた。他タスクに関しては一部の K で小型モデルよりも Top-K Jaccard 係数は高かった。同じ知識蒸留モデルでサイズが異なる TinyBERT-6L と TinyBERT-4L を比較すると、STS-B、MNLI、RTE に関しては、性能の高い TinyBERT-6L の方が Top-K Jaccard 係数が低い傾向が見られ、性能の高さと注意単語の一致度に乖離があることが確認できる。

以上から、先行研究での指摘の通り、知識蒸留モデルは小型モデルと比べて教師モデルと異なる単語に注目して推論を行うケースが数多くあることが確認された。知識蒸留は教師からパラメタ数が小さいモデルへ知識を転移させることで、単にパラメタ数を削減するだけの手法よりも高い精度を示すことが多い。しかし、精度が高いことに加えて、推論結果の根拠を示すことについても教師モデルと同等の能力を持ったモデルであることが望ましい。

第4章　注意単語を転移する知識蒸留

前章の結果から、知識蒸留モデルが様々なタスクに対して高い性能を示す一方で、知識蒸留を行っていないパラメータ数の少ない小型モデルと比較して、教師モデルとの間で推論根拠とする単語の一致度が低いケースが多いことがわかった。モデルが重要視する単語はモデルの出力の根拠を説明するものであるが、知識蒸留によって得られた生徒モデルも教師モデルと同程度の説明能力を有することが望ましい。

本章では、生徒モデルを学習する際に教師モデルと同じ単語に注意して推論を行うよう促す損失関数を導入し、これを知識蒸留の既存の損失関数に組み込むことで、精度と推論根拠の両面で教師モデルに近い振る舞いを示す生徒モデルを構築する手法を提案する。

4.1　提案手法

提案手法の概要を図 4.1 に示す。本手法は図中に示す Step1 と Step2 の 2 段階で構成される。

Step1 では、教師モデルと生徒モデルの双方に同一の学習データを入力し、推論結果に対して Integrated Gradients を適用することで、入力データにおける各単語の重要度スコアを算出する。このスコアはモデルが入力文中のどの単語を推論の根拠として重要視しているか (注意しているか) を表す指標である。それぞれのモデルについて、入力文中の全ての単語に対する重要度の分布 (以下、単に「重要度分布」と呼ぶ) を作成する。

Step2 では、Step1 で得られた 2 つの重要度分布間の整合性を高めるための損失関数を定義する。この損失を最小化するように生徒モデルを学習させることで、生徒モデルに教師モデルと同様の単語に注意して推論を行わさせることを狙う。

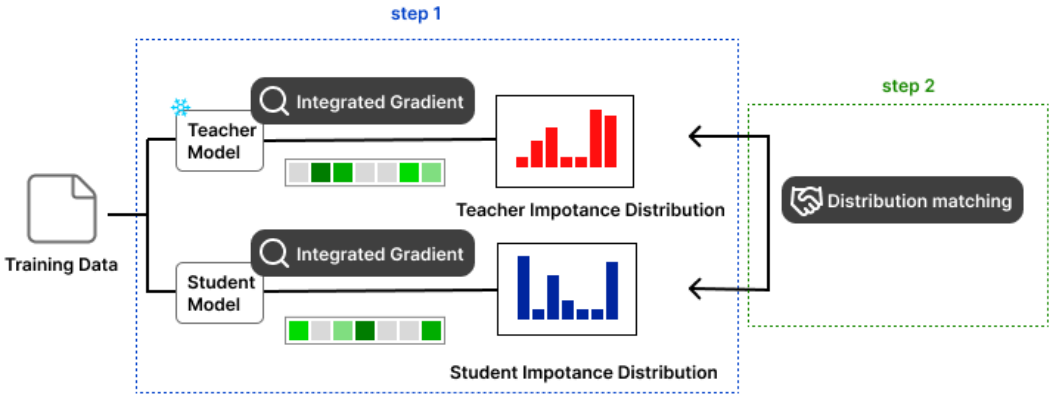


図 4.1: 提案手法の概要

本研究では、教師モデルとして下流タスクに対してファインチューニングされた BERT モデルを用いる。知識蒸留の手法としては TinyBERT を採用する。TinyBERT における知識蒸留は、2.3 節で述べたように、事前学習における知識蒸留、ファインチューニング時における知識蒸留の 2 段階で行われ、後者はさらに中間層蒸留と出力層蒸留の 2 段階に分けられる。ここでは事前学習の知識蒸留が完了した公開モデルを初期の生徒モデルとし、これを知識蒸留を行いつつファインチューニングする際、出力層蒸留において提案する損失関数を組み込む。

4.1.1 重要単語の抽出

教師モデルならびに生徒モデルの推論時における重要単語を抽出する方法について述べる。具体的には学習の過程において学習データの 1 つのインスタンス毎に Integrated Gradients を適用し、そのインスタンスにおけるそれぞれの単語の重要度スコアを算出する。SST-2 タスクにおけるインスタンスの例を以下に示す。SST-2 タスクは、与えられた英文が肯定的か否定的かを判定する二値分類タスクである。括弧の中はモデルが予測すべきラベルに対応する。この例では、予測ラベル「0」は negative(否定的) に対応する。

例
sentence: seem weird and distanced
label: negative (0)

このインスタンスに対しモデルが予測を行う際、入力はトークンに分割される。以下はトークン分割の結果である。

トークン列: [CLS] seem weird and distance ##d [SEP]

[CLS] と [SEP] は特殊トークンであり、トークナイザによって付加される。##d はサブワードである。サブワードとは、単語をさらに細かく分割したものであり、BERT のトークナイザでは未知語に対してサブワード分割が行われる。上記の例では、distanced が distance と##d に分割されている。

分割された各トークン (単語) に対し、Integrated Gradients を適用し、トークンの埋め込みベクトルの各次元に対して式 (2.9) に示す IG スコアを算出する。最終的に、埋め込みベクトルの各次元の IG スコアを集約することで各トークンの重要度スコアを算出する。BERT の埋め込みベクトルは 768 次元であるため、各トークンに対して 768 個の IG スコアを集約することになる。埋め込みの次元の IG スコアを集約する方法には様々なものがあるが、本研究では L2 ノルムを用いる。すなわち、インスタンス n におけるトークン t に対する重要度スコア $S_{n,t}$ を式 (4.1) で求める。

$$S_{n,t} = \|\mathbf{IG}_{n,t}\|_2 \tag{4.1}$$

ここで、$\mathbf{IG}_{n,t}$ はインスタンス n のトークン t における埋め込みベクトルの各次元に対する Integrated Gradients の値をまとめたベクトルを表す。BERT では埋め込みベクトルの次元数は $d = 768$ であり、$\mathbf{IG}_{n,t} \in \mathbb{R}^{768}$ となる。

具体例として、先に示した SST-2 のインスタンスに対し、ファインチューニングされた BERT モデルを用いて算出された IG スコアを図 4.2 に示す。

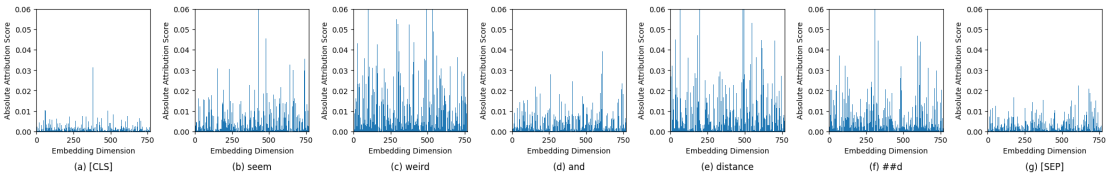


図 4.2: Integrated Gradients によって算出されたトークン埋め込みの各次元のスコア

図 4.2 の各トークンにおいて、横軸は埋め込みベクトルの次元を表し、縦軸は各次元に対する IG スコアを表す。L2 ノルムを用いて各トークンの IG 値を集約すると、図 4.3 のような重要度スコアが得られる。

図 4.3: L2 ノルムによって集約された各トークンの重要度スコア

このトークン列において一番重要であると評価されたのは「weird」であり、次いで「seem」と「distance」が重要であると評価されている。サブワードや特殊トークンにもスコアが割り振られているが、本手法ではこれらを削除したり無視したりせず、そのまま扱う。このように各インスタンス内のトークンの重要度スコアを算出することで、モデルがどの単語に注意して推論を行っているかを評価する。

4.1.2 重要度分布の構築

前項で算出した各トークンの重要度スコアを用いて、教師モデルと生徒モデルの重要度分布を作成する。重要度スコア $S_{n,t}$ は各トークンがモデルの予測に与える影響の大きさを表す非負のスカラー値である。本研究では、この重要度スコアをそのまま用いるのではなく、確率分布に変換してから知識蒸留のための損失関数を計算する。本節の冒頭で述べたように、トークンの重要度の確率分布を本研究では「重要度分布」と呼ぶ。

Integrated Gradients によって算出される重要度スコアの値の範囲は、インスタンスごとに大きく異なる可能性がある。絶対値では大きく異なっても、各インスタンス内での相対的な重要性は類似している場合がある。しかし、重要度スコアの絶対値をそのまま用いると、この相対的な類似性を正しく捉えることができない。この問題を解決するため、重要度スコアを確率分布に変換する。確率分布への変換では、各インスタンス内でのトークンの重要度スコアを正規化し、全トークンの確率の合計が 1 になるように調整する。これにより、重要度スコアの絶対的なスケールに依存せず、「各インスタンス内でどのトークンがどの程度重要か」という相対的な情報を表現できる。

具体的には、重要度分布は以下の手続きで作成する。

1. 重要度分布の温度スケーリングの導入

インスタンス n のトークン t における重要度スコア $S_{n,t}$ を温度パラメータ τ で除算する。

$$s_{n,t} = \frac{S_{n,t}}{\tau} \tag{4.2}$$

温度パラメータ τ は、重要度分布の鋭さを制御するハイパーパラメータである。$\tau > 1$ の場合、確率分布は平滑化され、より均一な確率分布に近づく。これにより、重要度が中程度のトークンにもある程度大きい確率が割り当てられる。一方、$\tau < 1$ の場合、確率分布は鋭くなり、重要度が高いトークンへの確率集中が強調される。$\tau = 1$ の場合は、スケーリングを行わず、元の重要度スコアをそのまま用いることに相当する。本研究では、予備実験において、いくつかの設定を試した結果、タスクによって温度を調整することが有効であると判断した。

2. ソフトマックス正規化

次に、スケーリング後のスコアをソフトマックス関数で正規化し、確率分布に変換する。教師モデルの重要度分布 $p^T_{n,t}$ は式 (4.3) で得られる。

$$p^T_{n,t} = \frac{\exp(s^T_{n,t})}{\sum_{t'=1}^{L} \exp(s^T_{n,t'})} \tag{4.3}$$

同様に、生徒モデルの重要度分布 $p^S_{n,t}$ は式 (4.4) で得られる。

$$p^S_{n,t} = \frac{\exp(s^S_{n,t})}{\sum_{t'=1}^{L} \exp(s^S_{n,t'})} \tag{4.4}$$

ここで、$s^T_{n,t} = S^T_{n,t}/\tau$ は教師モデルの温度スケーリング後の重要度スコア、$s^S_{n,t} = S^S_{n,t}/\tau$ は生徒モデルの温度スケーリング後の重要度スコアを表す。L はインスタンス n のトークン長を表す。この正規化により、$\sum_{t=1}^{L} p^T_{n,t} = 1$ かつ $\sum_{t=1}^{L} p^S_{n,t} = 1$ を満たす確率分布が得られる。

4.1.3 重要度分布を用いた損失関数

前項で計算した重要度分布はモデルがどの単語に注意しているかを示すものである。本研究では生徒モデルの重要度分布が教師モデルのそれと近くなるように、すなわち生徒モデルが教師モデルと同じような単語に注意するように知識蒸留を行う。本項ではそのための損失関数を計算する。基本的には、教師モデルと生徒モデルの重要度分布の差を定量化し、これをもとに損失関数を定義する。

本研究では、重要度分布の類似度を測る指標として Jaccard 係数を用いる。Jaccard 係数は、二つの集合の共通部分の大きさを和集合の大きさで割った値として定義され、集合間の重なり具合を評価する指標である。集合 A および B に対する Jaccard 係数は以下の (4.5) で定義される。

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{4.5}$$

Jaccard 係数は 0 から 1 の値を取り、二つの集合が完全に一致する場合に 1、共通部分を持たない場合に 0 となる。

しかしながら、従来の Jaccard 係数は、要素が「含まれる」または「含まれない」という離散的な集合に対して定義される指標である。そのため、Jaccard 係数は入力に対して不連続であり、微分が定義できない。この性質により、勾配降下法によってパラメータを更新するニューラルネットワークの損失関数として Jaccard 係数を直接用いることはできない。

そこで本研究では、Jaccard 係数を連続値を取る確率分布に拡張した Soft Jaccard Similarity [13] を用いる。Soft Jaccard Similarity は、従来の Jaccard 係数におけ

る集合の「共通部分」および「和集合」という概念を確率分布に対して転用し、これらの大きさを連続値として近似することで、微分可能な類似度指標として定義される。

具体的には、教師モデルの重要度分布を p^T、生徒モデルの重要度分布を p^S とすると、Soft Jaccard Similarity は式 (4.6) で定義される。

$$J_{\text{soft}}(p^T, p^S) = \frac{\sum_{t=1}^{L} p_{n,t}^T \cdot p_{n,t}^S}{\sum_{t=1}^{L} (p_{n,t}^T)^2 + \sum_{t=1}^{L} (p_{n,t}^S)^2 - \sum_{t=1}^{L} p_{n,t}^T \cdot p_{n,t}^S} \tag{4.6}$$

分子は二つの重要度分布の内積であり、従来の Jaccard 係数における共通部分の大きさに対応する。一方、分母はそれぞれの分布の二乗和から内積を引いたものであり、和集合の大きさに対応する量を近似している。Soft Jaccard Similarity は、集合に対する Jaccard 係数の基本的な考えを継承しつつ、2 つの確率分布に対する連続的かつ微分可能な類似度指標として機能する。

本研究では、教師モデルと生徒モデルの重要度分布間の Soft Jaccard Similarity を損失関数として用い、勾配降下法による学習を通じて、生徒モデルの重要度分布を教師モデルの重要度分布に近づける。具体的には、重要度分布の差を定量化した蒸留損失項 $\mathcal{L}_{\text{SJ}}$ を式 (4.7) と定義する。

$$\mathcal{L}_{\text{SJ}} = 1 - J_{\text{soft}}(\mathbf{p}^T, \mathbf{p}^S) \tag{4.7}$$

モデルを学習する際、類似度の最大化を目的とする代わりに、損失関数の最小化問題として扱うために損失項を $1 - J_{soft}$ の形で定義している。これにより、2 つの重要度分布が完全に一致している場合は $\mathcal{L}_{\text{SJ}} = 0$ となり、類似性が高い場合ほど損失が小さくなるという直観的な対応が得られる。本研究で提案する知識蒸留項は他の知識蒸留手法 (具体的には TinyBERT) に組み込むこと、すなわち交差エントロピー損失や既存の損失関数に加算して用いることを想定しているため、知識蒸留項を最小化問題に適用できるように設計している。

4.1.4 提案手法を用いた生徒モデルの学習

本研究では知識蒸留手法である TinyBERT を拡張し、元の TinyBERT で得られた生徒モデルの下流タスクに対する性能を損なうことなく、教師モデルと同じ単語に注意して推論を行う生徒モデルを学習することを目指す。既に述べたように、TinyBERT のファインチューニングは 2 段階で行われる。第 1 段階で中間層の隠れ状態や Attention 重みを教師モデルから模倣したモデルを学習し、第 2 段階では出力層の隠れ状態や Attention 重みを模倣する。本研究ではこの第 2 段階で重要度分布の差を定量化した新たな損失項を組み込む。

TinyBERT におけるファインチューニングの第 2 段階では、予測層蒸留損失 $\mathcal{L}_{\text{pred}}$ が用いられる。$\mathcal{L}_{\text{pred}}$ は、教師モデルのロジット（ソフトラベル）と生徒モデルのロジットの間の交差エントロピーであり、式 (4.8) で定義される。

$$\mathcal{L}_{\text{pred}} = \text{CE}(\mathbf{z}^T/t, \mathbf{z}^S/t) \tag{4.8}$$

ここで、$\mathbf{z}^S$ と $\mathbf{z}^T$ はそれぞれ生徒モデルと教師モデルが予測するロジット、CE はクロスエントロピー損失、t は温度パラメータを表す。TinyBERT の実験では $t = 1$ が有効であると報告されており、本研究でも同様に設定する。本研究では、提案する蒸留損失項を TinyBERT の予測層蒸留損失に加算し、最終的な損失関数を定義する。具体的な式を以下に示す。

$$\mathcal{L} = \mathcal{L}_{\text{pred}} + \lambda \mathcal{L}_{\text{SJ}} \tag{4.9}$$

ここで、λ は提案する蒸留損失項の重みを制御するハイパーパラメータである。λ の値を調整することで、予測クラスの一致度に対して注意単語の一致度をどれだけ重視するかを調整する。

第5章　評価実験

本章では、提案手法の有効性を評価するための評価実験について述べる。

5.1　実験設定

提案手法を以下のように実装する。教師モデルとしては、予め GLUE の各タスクでファインチューニングした BERT-base モデル[1]を用いる。生徒モデルの初期モデルとして、既存の知識蒸留手法である TinyBERT によって事前学習の段階で知識蒸留が行われ、公開されているモデルを用いる。具体的には、6 層の Transformer 層から構成される TinyBERT-6L[2]と 4 層の Transformer 層から構成されている TinyBERT-4L[3] 用いる。これらのモデルを初期パラメタとし、下流タスクのデータセットを用いてファインチューニングする際、提案手法の知識蒸留の手法を用いて生徒モデルを学習する。また、TinyBERT をベースラインとし、提案手法と性能を比較する。

本実験で比較する手法を以下にまとめる。

- **TinyBERT-6L**: TinyBERT の 6 層版の公開モデルを初期のモデルとし、従来の TinyBERT の蒸留手法で学習したモデル。
- **TinyBERT-4L**: TinyBERT の 4 層版の公開モデルを初期のモデルとし、従来の TinyBERT の蒸留手法で学習したモデル。
- **TinyBERT-6L_IGLoss**: TinyBERT の 6 層版の公開モデルを初期モデルとし、提案手法の損失関数を組み込んで学習したモデル。
- **TinyBERT-4L_IGLoss**: TinyBERT の 4 層版の公開モデルを初期モデルとし、提案手法の損失関数を組み込んで学習したモデル。

以降、TinyBERT-6L および TinyBERT-4L をベースラインとし、提案手法である TinyBERT-6L_IGLoss および TinyBERT-4L_IGLoss と比較する。

[1] https://huggingface.co/bert-base-uncased
[2] https://huggingface.co/huawei-noah/TinyBERT_General_6L_768D
[3] https://huggingface.co/huawei-noah/TinyBERT_General_4L_312D

データセット　実験にはGLUE データセットを用いる。対象タスクはGLUE データセットにおける 8 つのタスク、すなわち MRPC、CoLA、STS-B、RTE、QQP、MNLI、QNLI、SST-2 とする。TinyBERT の文献 [3] でも GLUE のデータセットを対象に評価実験を行っている。その際、GLUE のデータセットに対してデータ拡張を行い、訓練データ量を増やしている。具体的には、訓練データのサンプル中の単語をランダムにマスクし、それに当てはまる単語を BERT で予測し、その単語で置換することでデータを拡張している。また、データ拡張を行うコードも公開されている。本実験は TinyBERT をベースラインとするため、公平な評価のため、公開されているコードを用いてデータ拡張を行う。この際、拡張率 (元のデータに対してどれくらいサンプルを増やすか) はタスク毎に決められているが、本実験でもこの拡張率にしたがってデータ拡張を行う。ただし、QQP タスクと QNLI タスクについては拡張データの量が多く、本実験の計算機環境ではモデルの学習に時間がかかりすぎるため、拡張後のデータに対してランダムサンプリングを行いサンプル数を削減する。この際、ランダムサンプリング後のサンプル数が SST-2 タスクと同程度になるようにする。実験に用いた訓練データの統計を表 5.1 に示す。「調整後データ量」は QQP タスクと MNLI タスクではランダムサンプリング後のサンプル数を表し、これが実際に用いた訓練データのサンプル数となる。

表 5.1: 各タスクにおける訓練データの統計。「RS」はランダムサンプリングの実行の有無を表す。

タスク	学習データ数	拡張後データ量	RS	調整後データ量	テストデータ量
MRPC	3,669	225,098	無	225,098	1,725
CoLA	8,552	211,059	無	211,059	1,063
STS-B	5,750	320,113	無	320,113	1,379
RTE	2,491	68,796	無	68,796	3,000
QQP	363,847	7,572,066	**有**	1,107,510	390,965
MNLI-m	392,703	8,094,053	**有**	1,091,538	9,796
MNLI-mm	同上	同上	**有**	同上	9,847
QNLI	104,744	1,250,571	無	1,250,571	5,463
SST-2	67,350	1,107,510	無	1,107,510	1,821

MNLI は MNLI-m と MNLI-mm の 2 つのテストセットがあるため、合計 9 つのテストセットでベースラインならびに提案手法で蒸留されたモデルの性能を測定する。

モデル学習時のパラメータ　提案手法を学習する際のパラメータは以下の通りである。

- エポック数: 3
- バッチサイズ: 16

- 学習率: 5e-5
- 最適化手法: AdamW
- 重要度分布における温度パラメータ τ: 0.5 (CoLA, QQP, SST-2), 0.25 (STS-B, QNLI, MNLI, RTE)
- Integrated Gradients のステップ数 m: 20

評価指標 (タスクの性能)　蒸留したモデルの GLUE タスクにおける性能を評価する際には、GLUE の公式設定にしたがい、accuracy、Matthews correlation、Pearson correlation を評価指標として用いる。Pearson correlation は STS-B タスク、Matthews correlation は CoLA タスク、accuracy はそれ以外のタスクの評価指標である。各指標の定義を以下に述べる。

- Accuracy: 分類タスクにおける正解数の割合を示す指標である。

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.1}$$

  ここで、TP は真陽性、TN は真陰性、FP は偽陽性、FN は偽陰性を表す。

- Matthews correlation: 二値分類タスクにおける予測性能を評価する指標である。

$$\text{Matthews correlation} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{5.2}$$

- Pearson correlation: 回帰タスクにおける予測値と実測値の相関を評価する指標である。

$$\text{Pearson correlation} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{5.3}$$

  ここで、x_i は予測値、y_i は実測値、$\bar{x}$ は予測値の平均、$\bar{y}$ は実測値の平均を表す。

評価指標 (注意単語転移)　生徒モデルが教師モデルと推論時にどれだけ同じ単語に注意しているかを評価するため、独自の評価指標である Top-K Jaccard 係数と Top-K Ranking 指標を用いる。

Top-K Jaccard 係数は、教師モデルと生徒モデルが注意する単語の重なり具合を評価する指標である。教師モデルと生徒モデルのそれぞれについて、入力における各単語の重要度を計算し、その上位 K 個の単語集合を得る。次に、2 つの重要

度単語集合の類似度を Jaccard 係数を用いて測る。Jaccard 係数は、2 つの集合の共通部分の大きさを和集合の大きさで割ったものである。具体的な計算例を表 5.2 に示す。この表は、教師モデル、生徒モデルが注意する単語を重要度の順に 10 個抽出した結果と、それぞれの K における Top-K Jaccard 係数を示している。例えば、Top-3 では、共通する単語が 2 つ (distanced, and) であり、和集合の単語が 4 つ (weird, distanced, and, seem) であるため、Jaccard 係数は 0.5 となる。

表 5.2: Top-K Jaccard 係数の計算例

ランク	1	2	3	4	5	6	7	8	9	10
教師モデル	weird	distanced	and	seem	.	but	really	too	yet	!
生徒モデル	seem	and	distanced	weird	.	really	but	yet	too	!
Top-K Jaccard	0.000	0.000	0.500	1.000	1.000	0.714	1.000	0.778	1.000	1.000

テストデータにおける全てのサンプルにおいて上記の Top-K Jaccard 係数を算出し、その平均値を計算し、テストデータ全体に対するモデルの注意単語転移の正確さを表す指標とする。以下、単に「Top-K Jaccard 係数」と書いたときはこの平均値を指すものとする。

Top-K Jaccard 係数はある K の時点で共通する単語の割合を示すが、単語の重要度の順位が異なる場合でも同じ評価値を与える。つまり、重要度の順位は考慮されない。一方で、順位の微小な変動に影響されにくく、重要単語集合の大域的な一致度を安定して測定できる。

Top-K Ranking 指標は Jaccard 係数とは異なり、生徒モデルと教師モデルとで注意単語とその順序がどの程度一致しているかを評価する指標である。ある K の時点において、一番最初の単語から K 番目の単語までの順位が一致している時に 1、それ以外は 0 となる。具体的な計算例を表 5.3 に示す。

表 5.3: Top-K Ranking 指標の計算例

ランク	1	2	3	4	5	6	7	8	9	10
教師モデル	weird	distanced	and	seem	.	but	really	too	yet	!
生徒モデル	weird	and	distanced	seem	.	really	but	too	yet	!
Top-K Ranking	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

例えば、K が 1 の時点では、教師モデルと生徒モデルの 1 位の単語が一致しているため、Top-1 Ranking 指標は 1.0 となる。一方、K が 2 以降では、順位が一致していないため、Top-2 Ranking 指標は 0.0 となる。

テストデータにおける全てのサンプルにおいて上記の Top-K Ranking 指標を算出し、その平均値を計算する。以下、単に「Top-K Ranking 指標」と書いたときはこの平均値を指すものとする。

Top-K Ranking 指標は、ある K の時点で重要単語のリストの完全一致を評価するため、Top-K Jaccard 係数よりも厳しい指標である。一方で、教師の推論に重要である単語を厳密に模倣できているかを評価することができる。

5.2 下流タスクに対するモデルの性能評価

本節では教師モデル、ベースラインならびに提案手法による知識蒸留モデルの下流タスクに対する性能を評価する。表 5.4 に比較対象としたモデルの 9 つの下流タスクに対する評価結果を示す。太字はベースラインと提案手法による同じサイズの知識蒸留モデルのうち高い方の評価指標であることを示している。

表 5.4: 下流タスクに対する性能の比較

Model	**MRPC**	**CoLA**	**STSB**	**RTE**	**QQP**	**MNLI_m**	**MNLI_mm**	**QNLI**	**SST-2**
Metric	accuracy	matthews	pearson	accuracy	accuracy	accuracy	accuracy	accuracy	accuracy
BERT-base(FineTuned)	88.245	53.393	87.952	72.922	90.883	84.554	84.469	91.453	92.433
TinyBERT-6L	86.275	**48.724**	**87.701**	63.899	**90.873**	82.425	82.669	**90.335**	**91.743**
TinyBERT-6L_IGLoss	**86.765**	47.984	87.544	**66.007**	88.779	**82.946**	**83.018**	89.888	90.896
TinyBERT-4L	85.539	37.480	**87.407**	63.177	**90.124**	**81.803**	82.170	87.058	**90.940**
TinyBERT-4L_IGLoss	**85.784**	**38.323**	86.936	**67.509**	89.004	81.153	**82.594**	**87.453**	90.896

提案手法は概ねベースラインと同等の性能を示している。6 層のモデルについては、CoLA、STS-B、QQP、QNLI、SST-2 の各タスクについて、TinyBERT-6L_IGLoss は TinyBERT-6L よりも評価指標が低いが、その差は小さい。逆に MRPC、RTE、MNLI-m、MNLI-mm の各タスクについては TinyBERT-6L_IGLoss はベースラインを上回り、特に RTE タスクでは差が大きい。4 層のモデルについては、9 つのうち 5 つのタスクで TinyBERT-4L_IGLoss は TinyBERT-4L を上回り、6 層のモデルと同様に RTE タスクにおいては差が大きい。とはいえ、両者の差はおおむね小さい。以上から、本研究で提案する重要度分布の差を考慮した蒸留項の導入は知識蒸留後のモデルの下流タスクに対する性能を大きく損わないことが確認された。

5.3 注意単語の転移に対するモデルの評価

本節では提案手法の知識蒸留手法が生徒モデルと教師モデルとどの程度同じ単語に注意して推論を行っているかを評価する。

5.3.1 Top-K Jaccard 係数による注意単語の転移の評価

ベースラインとして採用している TinyBERT-6L および TinyBERT-4L の各タスクにおける Top-K Jaccard 係数を表 5.5 および表 5.6 に示す。提案手法を適用し

て学習した TinyBERT-6L_IGLoss および TinyBERT-4L_IGLoss の各タスクにおける Top-K Jaccard 係数を表 5.7 および表 5.8 に示す。6 層と 4 層のそれぞれの軽量モデルについて、ベースラインと提案手法の Top-K Jaccard 係数を可視化したグラフを図 5.1、図 5.2 と図 5.3、図 5.4 に示す。これらの図では、横軸が「Top-K」の K の値、縦軸は Top-K Jaccard 係数を示す。また、グラフの各点に提案手法とベースラインの Top-K Jaccard 係数の差を示す。

表 5.5: TinyBERT-6L(ベースライン) の Top-K Jaccard 係数

Task	Top-1	Top-2	Top-3	Top-4	Top-5	Top-6	Top-7	Top-8	Top-9	Top-10
CoLA	0.498	0.497	0.566	0.647	0.729	0.782	0.821	0.857	0.877	0.883
MNLI-m	0.356	0.357	0.369	0.393	0.416	0.437	0.458	0.480	0.501	0.520
MNLI-mm	0.212	0.250	0.290	0.320	0.350	0.379	0.410	0.436	0.464	0.489
MRPC	0.132	0.136	0.153	0.174	0.196	0.219	0.239	0.267	0.288	0.306
QNLI	0.288	0.267	0.279	0.296	0.314	0.334	0.350	0.367	0.382	0.400
QQP	0.212	0.234	0.275	0.309	0.347	0.375	0.403	0.430	0.458	0.489
RTE	0.133	0.196	0.229	0.260	0.283	0.296	0.318	0.348	0.372	0.382
SST-2	0.365	0.335	0.333	0.359	0.389	0.416	0.445	0.472	0.503	0.530
STS-B	0.171	0.195	0.240	0.283	0.329	0.373	0.411	0.448	0.483	0.519

表 5.6: TinyBERT-4L(ベースライン) の Top-K Jaccard 係数

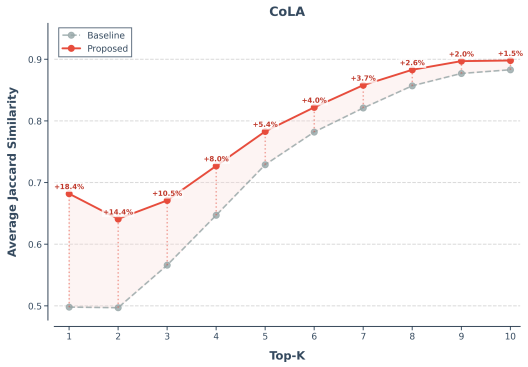
Task	Top-1	Top-2	Top-3	Top-4	Top-5	Top-6	Top-7	Top-8	Top-9	Top-10
CoLA	0.410	0.436	0.508	0.599	0.691	0.762	0.801	0.834	0.858	0.863
MNLI-m	0.341	0.355	0.382	0.412	0.442	0.470	0.494	0.518	0.540	0.560
MNLI-mm	0.164	0.211	0.242	0.271	0.297	0.324	0.348	0.371	0.394	0.423
MRPC	0.145	0.174	0.205	0.233	0.268	0.299	0.322	0.347	0.364	0.386
QNLI	0.245	0.233	0.247	0.267	0.289	0.311	0.329	0.347	0.365	0.383
QQP	0.212	0.238	0.277	0.315	0.349	0.377	0.403	0.428	0.454	0.481
RTE	0.162	0.222	0.266	0.286	0.306	0.335	0.356	0.386	0.410	0.422
SST-2	0.388	0.356	0.370	0.406	0.446	0.475	0.501	0.530	0.557	0.586
STS-B	0.191	0.228	0.275	0.314	0.360	0.408	0.453	0.503	0.548	0.585

表 5.7: TinyBERT-6L(提案手法) の Top-K Jaccard 係数

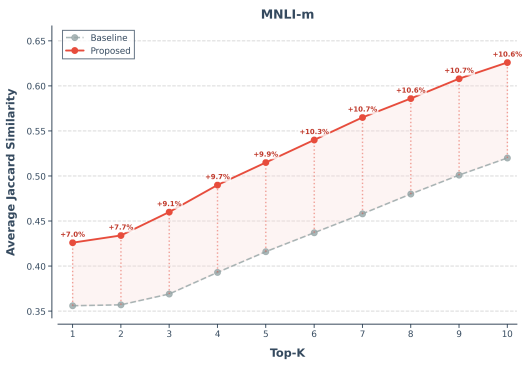
Task	Top-1	Top-2	Top-3	Top-4	Top-5	Top-6	Top-7	Top-8	Top-9	Top-10
CoLA	0.682	0.641	0.671	0.727	0.783	0.822	0.858	0.883	0.897	0.898
MNLI-m	0.426	0.434	0.460	0.490	0.515	0.540	0.565	0.586	0.608	0.626
MNLI-mm	0.324	0.390	0.444	0.489	0.522	0.451	0.481	0.508	0.532	0.556
MRPC	0.203	0.223	0.246	0.268	0.294	0.330	0.355	0.375	0.396	0.414
QNLI	0.327	0.303	0.321	0.339	0.360	0.379	0.397	0.414	0.433	0.451
QQP	0.439	0.461	0.494	0.527	0.555	0.579	0.597	0.616	0.635	0.658
RTE	0.199	0.230	0.264	0.295	0.325	0.343	0.364	0.387	0.414	0.432
SST-2	0.587	0.557	0.573	0.601	0.617	0.642	0.665	0.688	0.708	0.731
STS-B	0.311	0.327	0.365	0.417	0.466	0.508	0.554	0.597	0.633	0.665

表 5.8: TinyBERT-4L(提案手法) の Top-K Jaccard 係数

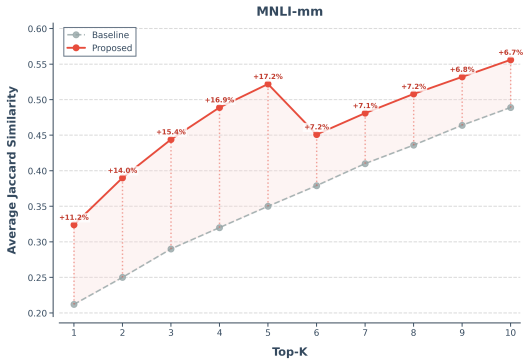
Task	Top-1	Top-2	Top-3	Top-4	Top-5	Top-6	Top-7	Top-8	Top-9	Top-10
CoLA	0.651	0.602	0.624	0.686	0.735	0.774	0.814	0.849	0.871	0.881
MNLI-m	0.450	0.453	0.481	0.508	0.534	0.560	0.581	0.601	0.619	0.636
MNLI-mm	0.319	0.392	0.444	0.482	0.512	0.544	0.473	0.501	0.526	0.551
MRPC	0.270	0.296	0.306	0.329	0.352	0.381	0.404	0.429	0.450	0.473
QNLI	0.331	0.322	0.332	0.355	0.379	0.399	0.418	0.436	0.452	0.468
QQP	0.406	0.435	0.471	0.508	0.537	0.561	0.579	0.597	0.615	0.635
RTE	0.260	0.304	0.335	0.381	0.388	0.415	0.439	0.447	0.478	0.491
SST-2	0.572	0.518	0.560	0.571	0.599	0.624	0.649	0.679	0.692	0.710
STS-B	0.265	0.291	0.337	0.390	0.438	0.482	0.526	0.567	0.607	0.641



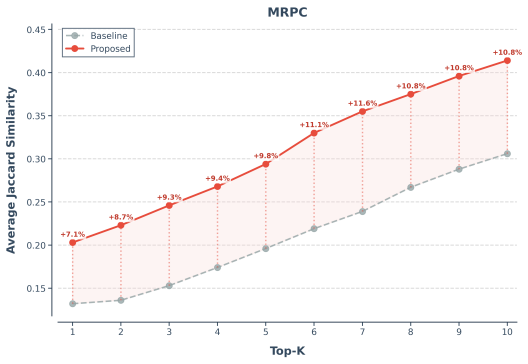
(a) CoLA



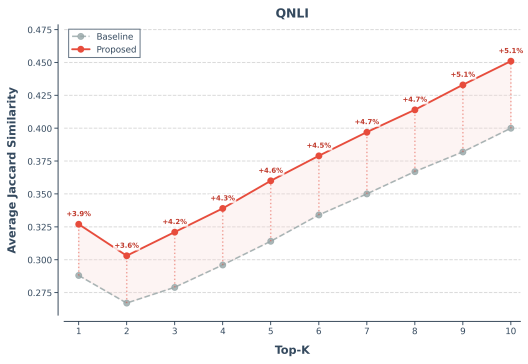
(b) MNLI-m



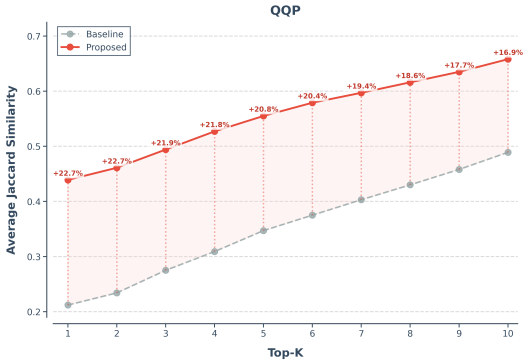
(c) MNLI-mm



(d) MRPC

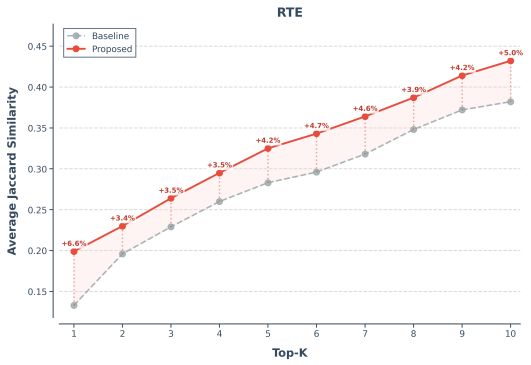


(e) QNLI

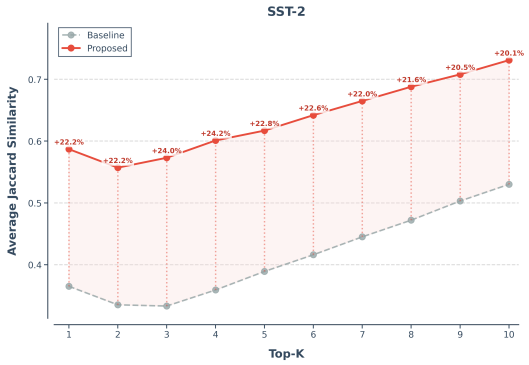


(f) QQP

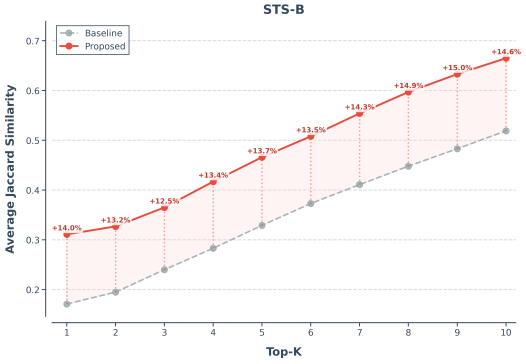
図 5.1: Top-K Jaccard 係数の比較 (TinyBERT-6L vs. TinyBERT-6L_IGLoss)



(a) RTE

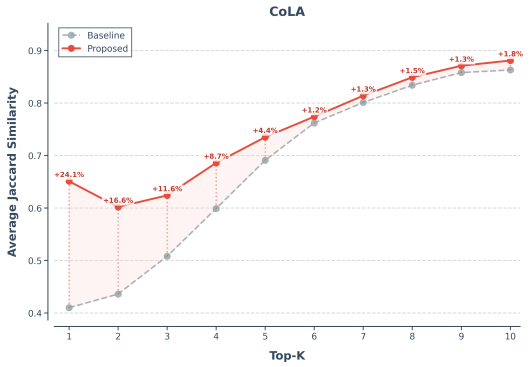


(b) SST-2

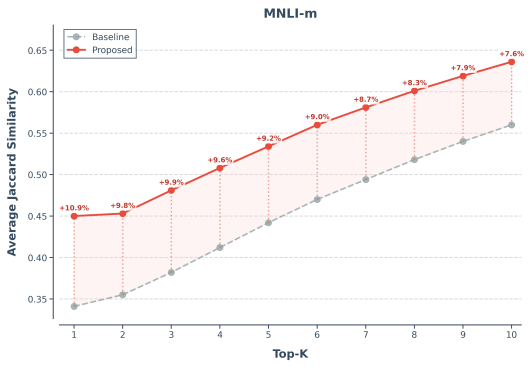


(c) STS-B

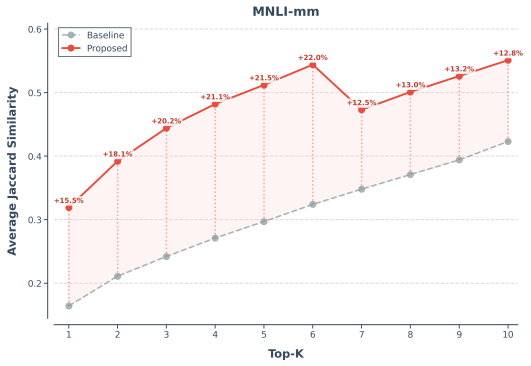
図 5.2: Top-K Jaccard 係数の比較 (TinyBERT-6L vs. TinyBERT-6L_IGLoss)(その 2)



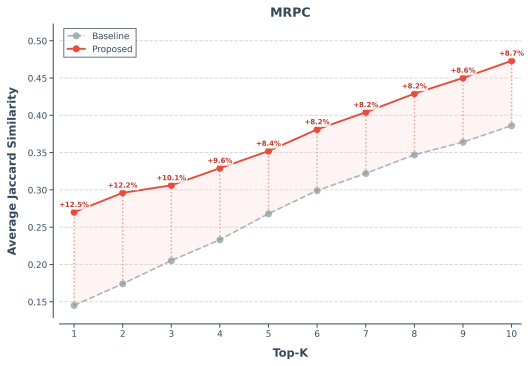
(a) CoLA



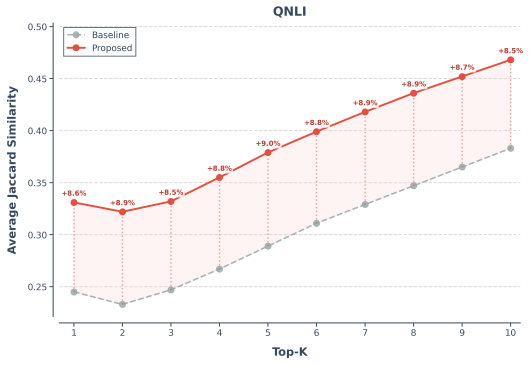
(b) MNLI-m



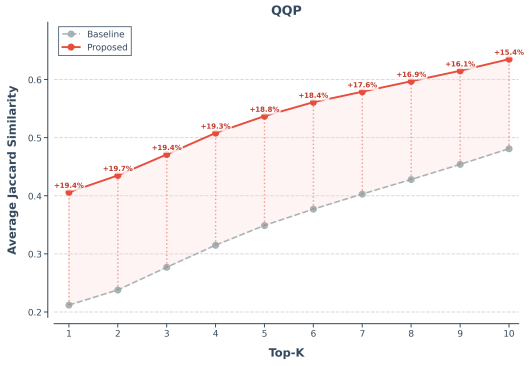
(c) MNLI-mm



(d) MRPC

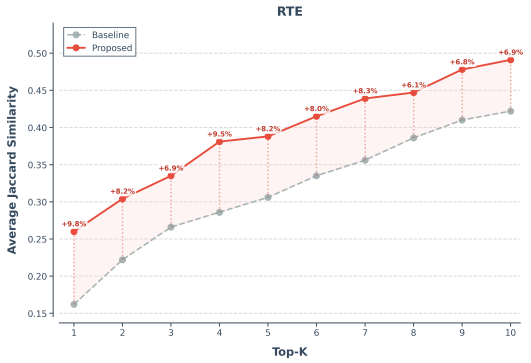


(e) QNLI

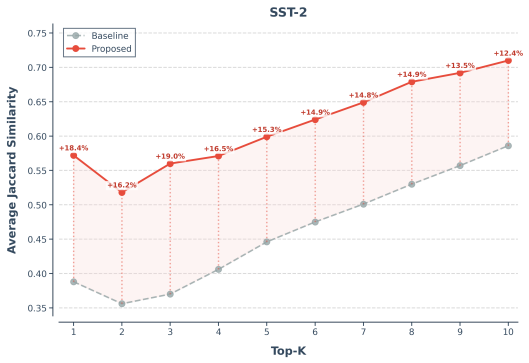


(f) QQP

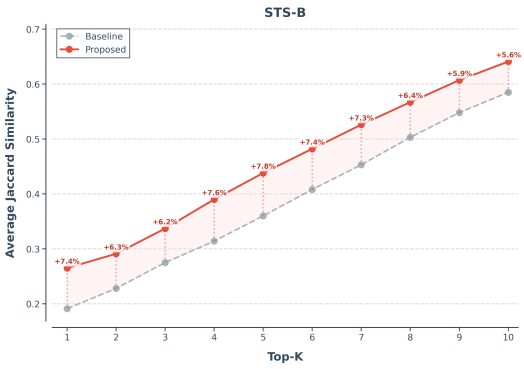
図 5.3: Top-K Jaccard 係数の比較 (TinyBERT-4L vs. TinyBERT-4L_IGLoss)



(a) RTE



(b) SST-2



(c) STS-B

図 5.4: Top-K Jaccard 係数の比較 (TinyBERT-4L vs. TinyBERT-4L_IGLoss)(その 2)

実験結果から、提案手法はベースラインと比べて、全タスクにおいて Top-K Jaccard 係数が向上した。特に、QQP と SST-2 に関して全体的に大きな改善が見られた。CoLA については、ベースラインの Top-K Jaccard 係数が比較的高く、特に Top-10 のような大きな K においては 0.8 程度であり、既存手法でも教師モデルと同じ単語に注目する生徒モデルが学習できていた。これらのケースでも差は小さいが提案手法の Top-K Jaccard 係数はベースラインを上回った。このことから、既存手法の TinyBERT で注意単語の転移が十分にできているときでもそうでないときでも、提案手法によって注意単語の転移を改善できることがわかった。

今回の実験に用いた GLUE データセットではタスクによってデータ量に差が見られる。表 5.1 に示したように、MRPC、CoLA、RTE は他のタスクに比べてデータ量が少ないが、これらのタスクにおいても提案手法によって Top-K Jaccard 係数が改善した。したがって、提案手法による注意単語転移のための蒸留項の導入は訓練データ量が十分に多くない場合でも有効に働くことがわかった。

図 5.1 - 図 5.4 に示したグラフの形状を見ると、特定の K のみに集中して Top-K Jaccard 係数の向上が見られるのではなく、全体的に均一に向上している様子が見

られる。モデルの推論の過程の説明のために注意単語を提示する際には、重要度の上位いくつの単語を提示するか (K の値) は状況によるが、どのような K に対しても提案手法によって学習した生徒モデルは教師モデルと同じ単語に注意を向ける傾向が強いと言える。

STS-B タスクについては、表 5.4 に示すように提案手法の正解率はベースラインと比べてわずかに低いが、Top-K Jaccard 係数は大きく改善している。本研究では注意単語転移のための蒸留項の重みを決めるハイパーパラメタ λ (式 (4.9) 参照) があるが、これを適切に調整することにより、タスクの正解率と注意単語の類似度を両方とも改善できる可能性がある。

5.3.2 Top-K Ranking 指標による注意単語の転移の評価

ベースラインモデルの TinyBERT-6L および TinyBERT-4L について、各タスクにおける Top-K Ranking 指標を表 5.9 および表 5.10 に示す。提案手法を適用して学習した TinyBERT-6L_IGLoss および TinyBERT-4L_IGLoss の各タスクにおける Top-K Ranking 係数を表 5.11 および表 5.12 に示す。6 層と 4 層のそれぞれの軽量モデルについて、ベースラインと提案手法の Top-K Ranking 指標を可視化したグラフを図 5.5 - 図 5.8 に示す。これらの図では、横軸が「Top-K」の K の値、縦軸は Top-K Ranking 指標を示す。また、グラフの各点に提案手法とベースラインの Top-K Ranking 指標の差を示す。

表 5.9: TinyBERT-6L(ベースライン) の Top-K Ranking 指標

Task	Top-1	Top-2	Top-3	Top-4	Top-5	Top-6	Top-7	Top-8	Top-9	Top-10
CoLA	0.498	0.216	0.118	0.081	0.070	0.066	0.065	0.065	0.065	0.065
MNLI-m	0.338	0.122	0.039	0.015	0.006	0.003	0.003	0.002	0.002	0.002
MNLI-mm	0.164	0.051	0.011	0.000	0.000	0.000	0.000	0.000	0.000	0.000
MRPC	0.118	0.032	0.012	0.005	0.000	0.000	0.000	0.000	0.000	0.000
QNLI	0.288	0.064	0.014	0.002	0.001	0.000	0.000	0.000	0.000	0.000
QQP	0.212	0.108	0.039	0.023	0.008	0.004	0.001	0.000	0.000	0.000
RTE	0.134	0.051	0.011	0.004	0.000	0.000	0.000	0.000	0.000	0.000
SST-2	0.365	0.091	0.022	0.008	0.005	0.003	0.003	0.003	0.003	0.003
STS-B	0.171	0.065	0.022	0.009	0.002	0.000	0.000	0.000	0.000	0.000

表 5.10: TinyBERT-4L(ベースライン) の Top-K Ranking 指標

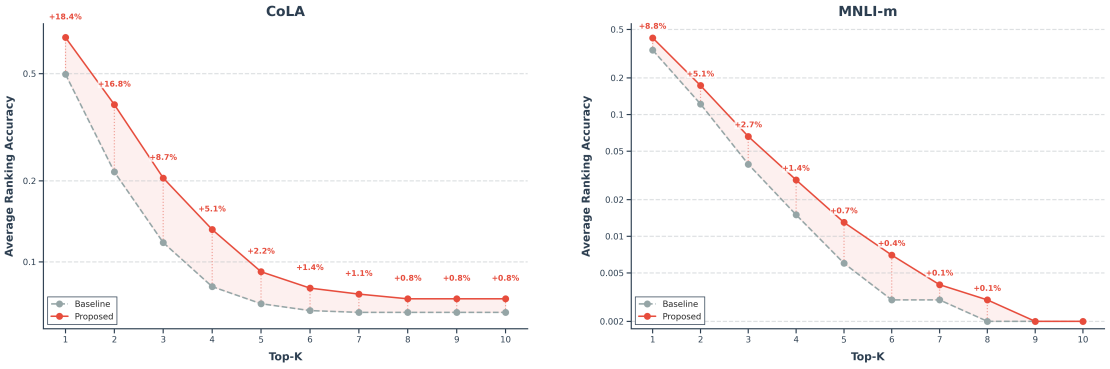
Task	Top-1	Top-2	Top-3	Top-4	Top-5	Top-6	Top-7	Top-8	Top-9	Top-10
CoLA	0.410	0.152	0.069	0.042	0.030	0.028	0.028	0.028	0.028	0.028
MNLI-m	0.341	0.113	0.039	0.015	0.007	0.003	0.002	0.001	0.001	0.001
MNLI-mm	0.273	0.075	0.017	0.000	0.000	0.000	0.000	0.000	0.000	0.000
MRPC	0.145	0.039	0.002	0.002	0.000	0.000	0.000	0.000	0.000	0.000
QNLI	0.245	0.046	0.007	0.002	0.000	0.000	0.000	0.000	0.000	0.000
QQP	0.212	0.086	0.029	0.014	0.005	0.002	0.000	0.000	0.000	0.000
RTE	0.162	0.043	0.018	0.007	0.000	0.000	0.000	0.000	0.000	0.000
SST-2	0.388	0.110	0.041	0.016	0.009	0.008	0.008	0.007	0.007	0.007
STS-B	0.191	0.079	0.022	0.012	0.005	0.001	0.000	0.000	0.000	0.000

表 5.11: TinyBERT-6L(提案手法) の Top-K Ranking 指標

Task	Top-1	Top-2	Top-3	Top-4	Top-5	Top-6	Top-7	Top-8	Top-9	Top-10
CoLA	0.682	0.384	0.205	0.132	0.092	0.080	0.076	0.073	0.073	0.073
MNLI-m	0.426	0.173	0.066	0.029	0.013	0.007	0.004	0.003	0.002	0.002
MNLI-mm	0.320	0.178	0.126	0.110	0.104	0.101	0.100	0.000	0.00	0.000
MRPC	0.203	0.056	0.007	0.002	0.000	0.000	0.000	0.000	0.000	0.000
QNLI	0.327	0.079	0.021	0.003	0.001	0.000	0.000	0.000	0.000	0.000
QQP	0.439	0.225	0.102	0.051	0.023	0.010	0.003	0.001	0.000	0.000
RTE	0.199	0.047	0.007	0.000	0.000	0.000	0.000	0.000	0.000	0.000
SST-2	0.587	0.280	0.112	0.052	0.019	0.010	0.008	0.008	0.008	0.008
STS-B	0.311	0.106	0.038	0.019	0.010	0.003	0.002	0.001	0.001	0.001

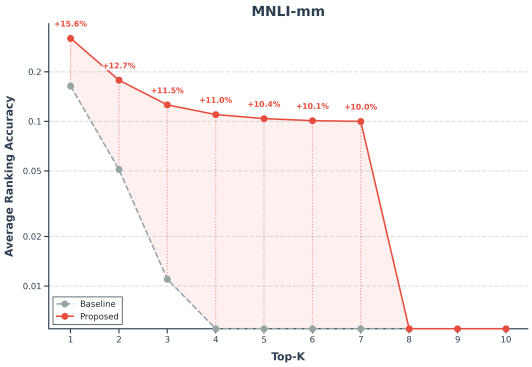
表 5.12: TinyBERT-4L(提案手法) の Top-K Ranking 指標

Task	Top-1	Top-2	Top-3	Top-4	Top-5	Top-6	Top-7	Top-8	Top-9	Top-10
CoLA	0.651	0.322	0.168	0.096	0.069	0.060	0.058	0.058	0.058	0.058
MNLI-m	0.450	0.187	0.080	0.033	0.015	0.007	0.004	0.003	0.003	0.002
MNLI-mm	0.319	0.175	0.125	0.108	0.102	0.101	0.098	0.000	0.000	0.000
MRPC	0.270	0.091	0.029	0.015	0.005	0.005	0.002	0.000	0.000	0.000
QNLI	0.331	0.096	0.021	0.004	0.001	0.000	0.000	0.000	0.000	0.000
QQP	0.406	0.205	0.088	0.045	0.019	0.008	0.003	0.001	0.000	0.000
RTE	0.260	0.090	0.025	0.007	0.004	0.000	0.000	0.000	0.000	0.000
SST-2	0.572	0.234	0.091	0.034	0.017	0.009	0.008	0.008	0.007	0.007
STS-B	0.265	0.086	0.027	0.013	0.007	0.003	0.003	0.001	0.001	0.001

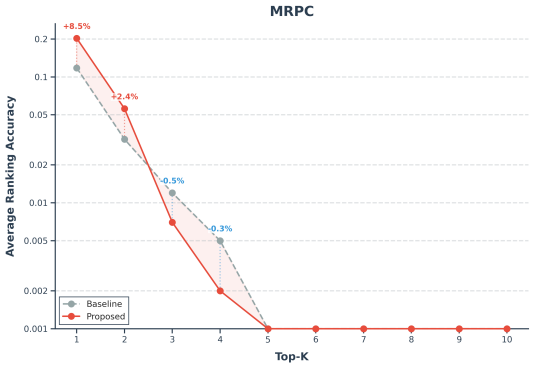


(a) CoLA

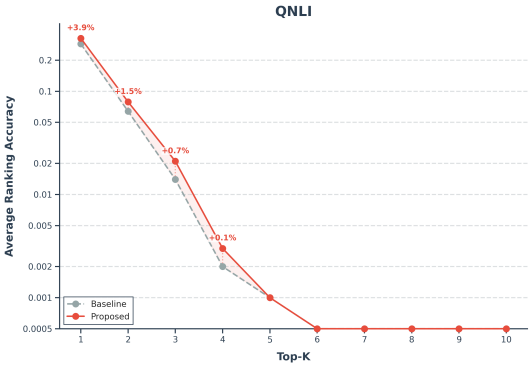
(b) MNLI-m



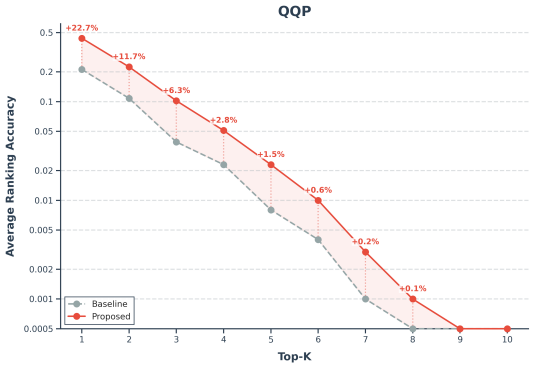
(c) MNLI-mm



(d) MRPC

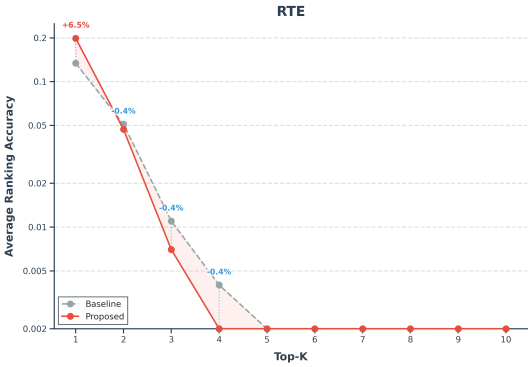


(e) QNLI

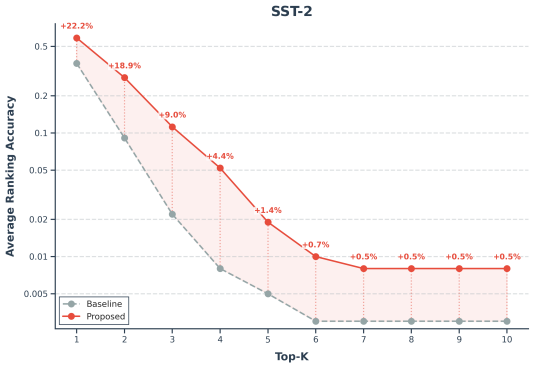


(f) QQP

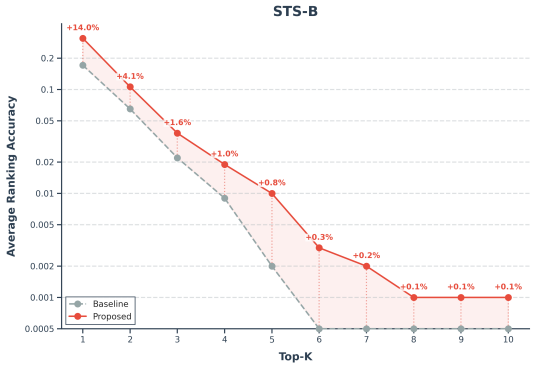
図 5.5: Top-K Ranking 指標の比較 (TinyBERT-6L vs. TinyBERT-6L_IGLoss)



(a) RTE

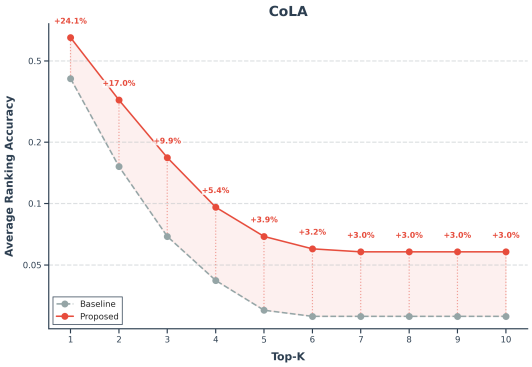


(b) SST-2

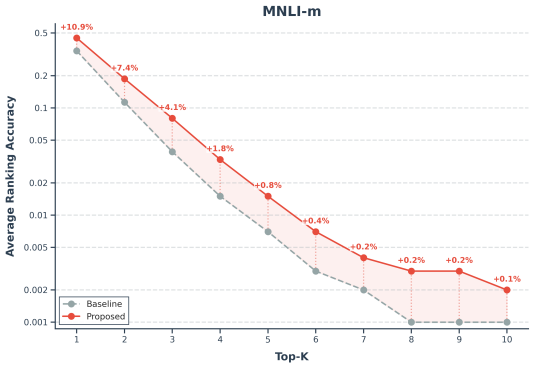


(c) STS-B

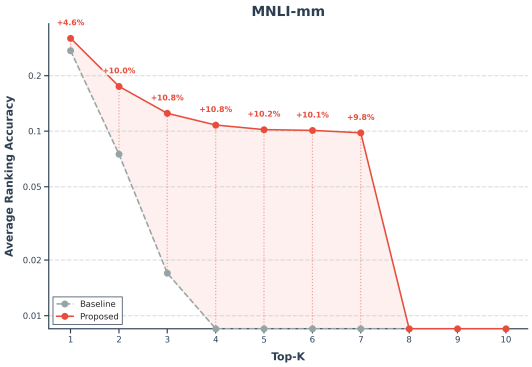
図 5.6: Top-K Ranking 指標の比較 (TinyBERT-6L vs. TinyBERT-6L_IGLoss)(その 2)



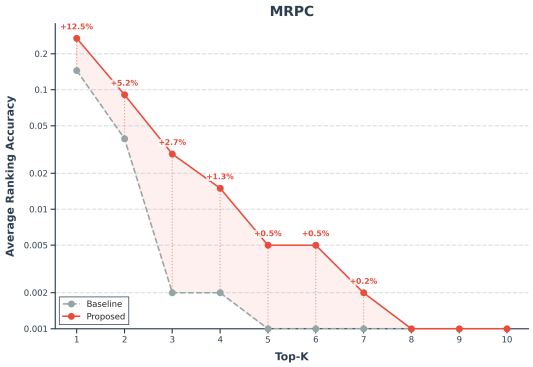
(a) CoLA



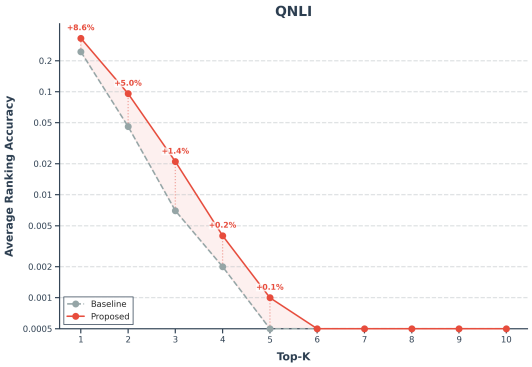
(b) MNLI-m



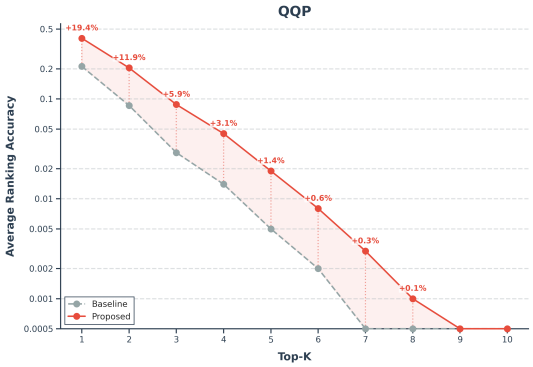
(c) MNLI-mm



(d) MRPC

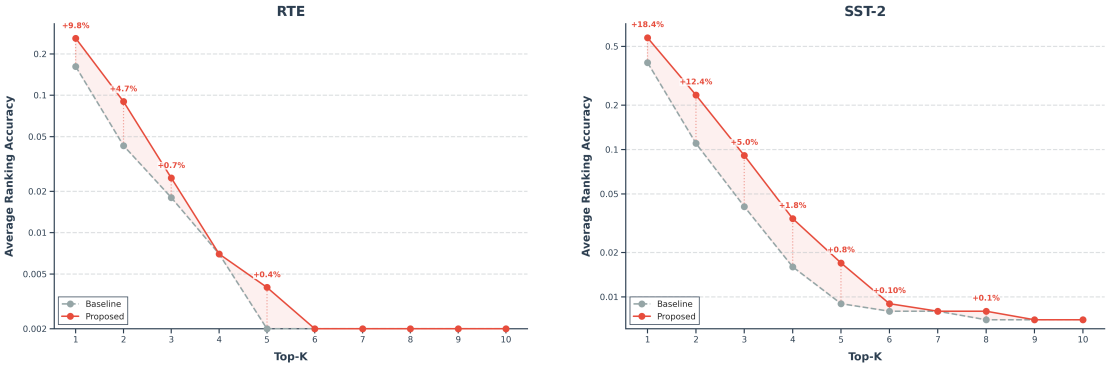


(e) QNLI



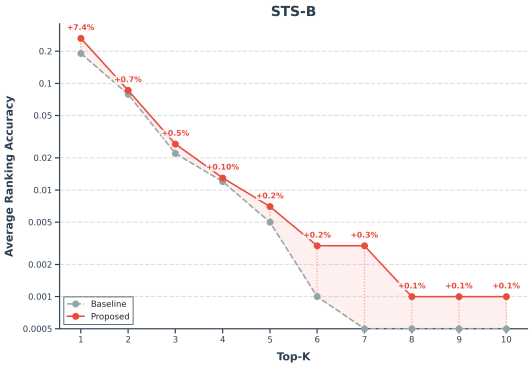
(f) QQP

図 5.7: Top-K Ranking 指標の比較 (TinyBERT-4L vs. TinyBERT-4L_IGLoss)



(a) RTE

(b) SST-2



(c) STS-B

図 5.8: Top-K Ranking 指標の比較 (TinyBERT-4L vs. TinyBERT-4L_IGLoss)(その 2)

Top-K Ranking 指標に基づく評価結果を分析すると、提案手法による改善の特徴が Top-K Jaccard 係数とは異なる形で現れていることがわかった。

まず、Top-1（最も重要な単語の一致）において、提案手法は顕著な改善を示した。具体的には、6L モデルにおいて、CoLA では 0.498 から 0.682 へ、SST-2 では 0.365 から 0.587 へ、QQP では 0.212 から 0.439 へと大幅に向上した。4L モデルでも同様に、CoLA で 0.410 から 0.651 へ、MRPC で 0.145 から 0.270 へ、RTE で 0.162 から 0.260 へと改善が見られた。このことは、提案手法が「最も重要な単語を正しく特定する」という点で特に効果的であることを示している。

次に、改善率の観点から 6L と 4L を比較すると、MRPC タスクでは、6L の改善率が 72%の改善であるのに対し、4L では 86%の改善であった。同様に RTE タスクでも、6L が 48%の改善、4L が 60%の改善と、4L の方が高い改善率を示した。パラメータ数が少ない 4L モデルの方が、提案手法によって注意単語を転移しやすいと考えられる。一方、教師モデルと生徒モデルのパラメタ数のギャップが大きい場合、教師モデルから生徒モデルへの知識の転移がうまくできないことが知られているが [5]、本実験では注意単語の転移について、より少ないパラメタ数を持つ 4L の方が改善の度合が大きいことは興味深い結果である。

第6章　おわりに

6.1　本研究のまとめ

本研究では、従来の知識蒸留では考慮されていなかった推論時における注意単語の教師モデルから生徒モデルへの転移に着目し下流タスクに対する推論能力だけでなく、モデルの推論における根拠となる単語を生徒モデルに転移させる手法を提案した。教師モデルと生徒モデルのそれぞれについて、訓練データのインスタンスを入力したとき、入力における各単語の重要度を Integrated Gradients を用いて算出した。具体的には、トークンの埋め込みの次元毎に Integrated Gradients によって重要度スコアを求め、そのスコアを値とする重要度ベクトルを作成し、その L2 ノルムを単語の重要度とした。次に、温度付きの Softmax 関数を用いて入力単語の重要度スコアの和が 1 になるように正規化し、訓練データのインスタンスにおける重要度分布 (単語の重要度の確率分布) を得た。そして、教師モデルと生徒モデルの重要度分布の差を定量化した損失項を設計した。この損失項を既存の知識蒸留手法である TinyBERT の蒸留損失項に加算し、新たな損失関数を設計した。この損失関数を最小化するように生徒モデルを学習することで、教師モデルが注意する単語の傾向を生徒モデルに継承した。

実験では、Top-K Jaccard 係数および Top-K Ranking 指標を評価指標として、GLUE の各における 9 つのタスクの評価セットにおいて、提案手法を適用した生徒モデルと既存の知識蒸留手法 TinyBERT によって学習した生徒モデルを比較した。これらの指標は、教師モデルと生徒モデルから重要度が高い上位 K 個の単語をそれぞれ抽出したとき、Top-K Jaccard 係数は重要単語の集合が一致しているか、Top-K Ranking 指標は K 個の重要語とその順位が一致しているかを測るものである。その結果、提案手法を適用した生徒モデルは、Top-K Jaccard 係数に関しては全てのタスクで TinyBERT を上回った。特に MRPC タスクと RTE タスクに関しては、タスク自体に対するモデルの性能の向上も見られた。他のタスクに関しては軽微な性能の低下が見られたものの、Top-K Jaccard 係数は大きく改善し、モデルの推論能力を大きく損わずに注意単語を転移することができた。これらのタスクに関しては、提案手法のハイパーパラメータを調整することで、モデルの性能の低下を抑えつつ注意単語の一致度を高めるように調整できる可能性がある。

Top-K Ranking 指標については、K を 1 から 10 まで変動させて算出したところ、一部の K においては提案する損失項の導入により低下したものの、多くの K においては改善し、全体的にはどのタスクにおいても Top-K Ranking 指標が改善する

傾向が確認できた。Top-K Ranking 指標は生徒モデルと教師モデルが注意する単語を順位も含めて一致するかを測るため、Top-K Jaccard 係数よりも厳密で難しい指標であり、K が大きくなるほど一気に低下しやすい。それにもかかわらず、多くのタスクにおいて K が大きい場合でも小さくはあるが改善が見られ、小さい K では大きく向上したことから、提案手法が教師モデルと生徒モデルの注意単語の一致度を高めることに寄与していることが示された。

本研究の貢献は、従来では重要視されていなかったモデルの推論過程の転移を考慮した知識蒸留のアプローチを提案した点にある。従来の知識蒸留手法では、モデルの性能にのみ着目し、推論過程における根拠の転移を考慮していなかった。知識蒸留は教師モデルの代替となるような小さなモデルを構築する技術であるが、知識蒸留の結果得られる生徒モデルは教師モデルと推論能力以外のあらゆる点での機序が一致することが望ましい。本研究はこの実現に向けた研究のひとつと位置付けられる。

6.2 今後の課題

評価実験では提案手法の一定の効果が確認されたものの、いくつかの課題が残された。

第一に、注意単語を転移するための損失項の改善である。本研究では、教師モデルと生徒モデルの重要度分布の Soft Jaccard 係数を最大化するように損失を設計したが、Soft Jaccard 係数の最大化が必ずしも注意単語の順位を考慮したより厳密な Top-K Ranking 指標の改善に寄与しなかった。人間にわかりやすく説明がしやすいモデルを構築するためには、単に注意単語の集合が一致するだけでなく、注意単語の順位も一致させることが望ましい。今後の課題として、Top-K Ranking 指標を直接最大化するような損失関数の設計が挙げられる。

第二に、Integrated Gradients におけるステップ数の調整にも課題が残る。ステップ数は計算時間と積分誤差の許容量のトレードオフによって決定するハイパーパラメータであり、本研究では経験的に 20 と設定した。しかしながら、近年の研究では、Integrated Gradients による最適なステップ数はインスタンスごとに異なることが指摘されており、その最適化の方法として、特定の誤差を下回るまでステップ数を増やしながら誤差を減らすアプローチが推奨されている [14]。本研究では、すべてのインスタンスに対して同じステップ数を利用しているが、今後の課題として、インスタンスごとに最適なステップ数を決定するアプローチを検討することが挙げられる。

参考文献

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[2] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

[3] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. TinyBERT: Distilling BERT for natural language understanding. In Trevor Cohn, Yulan He, and Yang Liu, editors, *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 4163–4174, Online, November 2020. Association for Computational Linguistics.

[4] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, p. 4768–4777, Red Hook, NY, USA, 2017. Curran Associates Inc.

[5] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant. *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34, No. 04, pp. 5191–5198, Apr. 2020.

[6] Rohit Raj Rai, Chirag Kothari, Siddhesh Shelke, and Amit Awekar. Compressed models are not trust-equivalent to their large counterparts, 2025.

[7] Marco Ribeiro, Sameer Singh, and Carlos Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In John DeNero, Mark Finlayson, and Sravana Reddy, editors, *Proceedings of the 2016 Conference of*

the North American Chapter of the Association for Computational Linguistics: Demonstrations, pp. 97–101, San Diego, California, June 2016. Association for Computational Linguistics.

[8] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[9] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks, 2017.

[10] Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Well-read students learn better: The impact of student initialization on knowledge distillation. *CoRR*, Vol. abs/1908.08962, , 2019.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[12] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In Tal Linzen, Grzegorz Chrupała, and Afra Alishahi, editors, *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics.

[13] Zifu Wang, Xuefei Ning, and Matthew B. Blaschko. Jaccard metric losses: Optimizing the jaccard index with soft labels, 2024.

[14] 牧野雅紘, 浅妻佑弥, 佐々木翔大, 鈴木潤. Integrated gradients における理想の積分ステップ数はインスタンス毎に異なる. 言語処理学会第 30 回年次大会, 2024.