令和 5 年度茨城大学工学部情報工学科卒業研究論文

LLM を利用した文書分類のための DataAugmentation

所属　情報工学科

著者　小野寺優 (20T4022L)

指導教員　新納浩幸教授

令和 6 年 2 月 2 日 (金)

令和 5 年度茨城大学工学部情報工学科卒業研究論文

LLM を利用した文書分類のための DataAugmentation

著者

小野寺優 (20T4022L)

指導教員

新納浩幸教授

論文要旨

大規模言語モデル (Large Language Model: LLM) は，膨大な量のテキストデータから学習された言語モデルである．代表的な LLM の例としては，GPT-3.5 や GPT-4 が挙げられる．LLM は様々な自然言語処理タスクに利用され，多くの成果を挙げている．また同時に LLM の高い文生成能力を生かして様々な利用方法が検討されている．

本研究では，LLM を利用した文書分類タスクのための訓練データの自動構築を試みる．具体的には，Data Augmentation と呼ばれるアプローチを取る．LLM で Data Augmentation を行う場合，少数のラベル付きデータからどのようなプロンプトで目的となる文書を生成すればよいかが重要となる．LLM を用いた Data Augmentation 手法は広く研究されているが，いずれの手法も短い文に特化した few-shot プロンプトによるデータ生成を行っており，長文データに対する有効性は明らかになっていない．しかし，LLM に入力できるトークン数は制限されており，長文をそのまま few-shot で与えるのは困難な場合が多い．そこで本研究では，キーワードと要約文の 2 要素で構成されたプロンプトを 3 種類作成し，長文データに対応した Data Augmentation を試みる．

実験では，livedoor ニュースコーパスを用いて訓練データの構築を行い，LLM により生成した拡張データを水増しした．これにより作成した拡張訓練データを利用して日本語 RoBERTa を使用した文書分類を行うことで，生成した拡張データの評価を行った．考察では，実験の結果と追加実験の実施により LLM を用いて文書分類に対するデータを自動構築する上での課題を分析した．

目次

表目次

図目次

第 1 章

序論

大規模言語モデル (Large Language Model: LLM) は，膨大な量のテキストデータから学習された言語モデルである．LLM は，質問応答，文章要約，あるいは機械翻訳など様々な自然言語処理タスクに利用され，多くの成果を挙げている．代表的な LLM の例には GPT-3.5 や GPT-4 [3] があり，高精度で自然な文章生成が可能である．そのため，LLM の能力を生かした多種多様な利用方法が検討され始めている．

本研究では，LLM の利用の一つとして文書分類タスクに対する訓練データの自動構築を試みる．具体的には，Data Augmentation のアプローチを取る．Data Augmentation とは，少数の訓練データを元に新たなデータを生成し，元のデータに水増しすることでモデルの性能を向上させることを目指す手法である．つまり，本研究では少数の訓練データとなる文書を元に LLM を利用して各分類ラベルに対する文書を生成し，そのラベル付き文書を元の少数の訓練データに追加することで分類精度を向上させることを目指す．

LLM を用いた Data Augmentation の手法はいくつか提案されている．Yoo ら [4] は，文とラベルの組から構成されたプロンプトを用いて，文とラベルを同時に LLM に自動生成させる手法を提案した．また，Sahu ら [1] は，特定のラベルに属する文から few-shot プロンプトを作成し，Data Augmentation を行うアプローチを提案した．しかし，これらの手法で扱われるデータは短い文のみであり，ニュース記事のような長文データに対する有効性は明らかになっていない．

長文データから few-shot プロンプトを作成する場合の問題の一つにトークン数がある．トークンとは，テキストデータを処理する際に用いられる基本的な単位のことを表す．トークン化の手法は言語モデルによって異なるが，GPT-3.5 の場合「私は犬が好き

です.」という文をトークン化すると 10 トークンに分割できる[*1] (図 1.1 参照). 一般的に LLM には，入力できるトークン数の上限が決められているため，トークン数が多い長文データを複数例示したプロンプトは入力が制限されることが多い．したがって，長文データに対して LLM による Data Augmentation を行う場合，如何にトークン数を抑えてラベル付きデータを利用したプロンプトを作成するのかがポイントとなる．

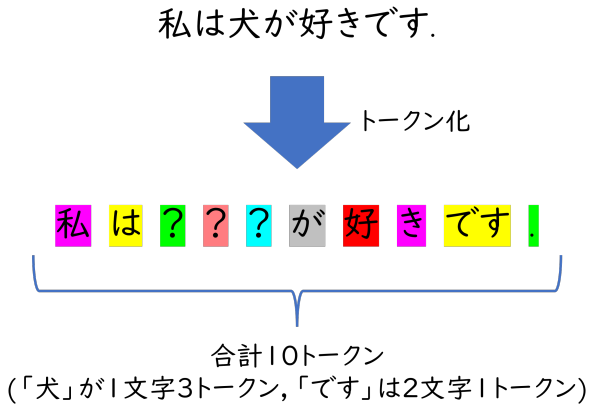

図 1.1: GPT3.5 におけるトークン化の例

本研究では，キーワードと要約文の 2 つから構成されたプロンプトにより，入力トークン数を抑え，長文にも対応可能な Data Augmentation 手法を試みる．キーワードは各分類ラベルの文書から LLM を用いて抽出することで獲得し，要約文は訓練データとなる文書を LLM に要約させることで生成する．キーワードや要約文の与え方を変えた 3 種類のプロンプトを作成し新たな文書を生成することで，生成文が文書分類に有効な文書データであるのかについて調査を行う．

本研究では livedoor ニュースコーパスからデータセットの構築を行い，文書分類タスクを実施した．文書分類のための分類モデルの構築には，rinna 社から公開されている日本語 RoBERTa を利用した．これの fine-tuning にデータを拡張していない訓練データを用いた場合の精度と 3 種類の試みたプロンプトにより Data Augmentation を行った拡張訓練データを用いた場合の精度を比較することで拡張データの有効性を評価する．

本論文では，上記実験結果を報告するとともに，LLM を用いて文書分類に対するラベル付きデータを生成する上での課題を考察する．

[*1] https://platform.openai.com/tokenizer

第 2 章

関連研究

2.1 言語モデル

自然言語処理 (Natural Language Processing：NLP) の分野において使用されるモデルの一種に言語モデルがある．これは，ある文章や単語列が与えられたときに次に来る単語やフレーズを出現確率を割り当てることで予測するモデルである．

例えば，「好きな動物は？」という文章に対しては，「犬」や「ネコ」といった単語には高い確率が割り当てられるが，「トマト」や「自転車」といった単語が割り当てられる確率は低くなる (図 2.1 参照)．

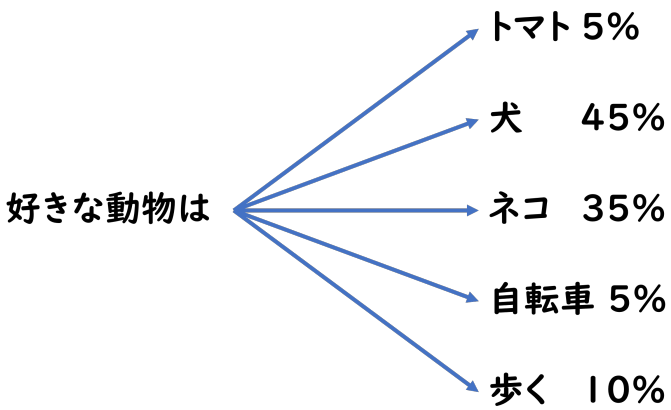

図 2.1: 言語モデルのイメージ

近年では，大量のテキストデータとディープラーニングの技術を用いて構築した大規模言語モデル (LLM) が登場し，大きな発展が見られた．LLM は様々な NLP タスクに利用され，多くの成果を挙げている．

2.1.1　Transformer

Transformer [5] は，2017 年に発表されたディープラーニングモデルである．当時，自然言語処理の分野で主流であった再帰構造 (RNN) や畳み込み構造 (CNN) を使用せず，Attention 機構のみを利用した Encoder-Decoder モデルである．Transformer は，機械翻訳タスクにおいて当時の SOTA[*1]を超える BLEU スコア[*2]を記録した．更に，RNN や CNN を用いたモデルでは困難であった並列計算が可能になり，訓練時間の短縮を実現させた．Transformer モデルの構造を図 2.2 に示す．

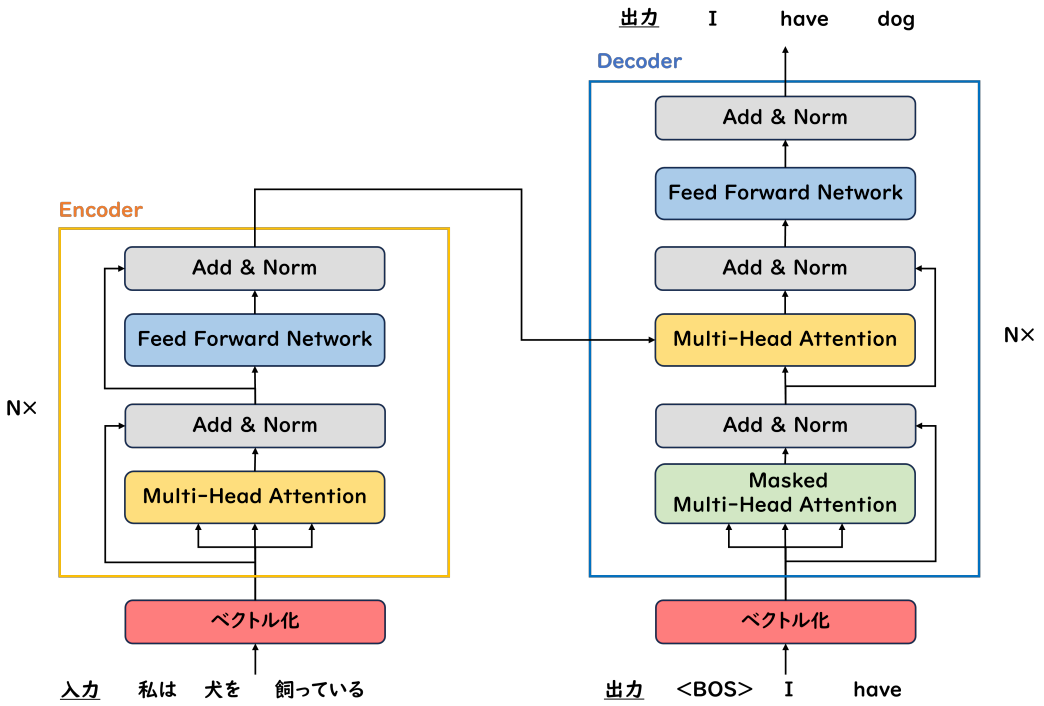

図 2.2: Transformer の構造

Transformer モデルにおける Attention 機構は Self-Attention という仕組みを使っている．これは，Transformer が文内の異なる位置の単語同士の関連性を計算するために使用される．Attention には，Query，Key，Value の 3 つの入力がある．検索にかけたい単語を Query としたとき，Query に最も近い Key を求め，Key を参考にして対応する Value を出力する．Transformer における Self-Attention では，Query を Q，Key を K，Value を V，Key ベクトルの次元数を d_k としたとき，Scaled Dot-Product Attention

[*1] ”State-of-the-Art” の略であり，現時点での最先端の性能を達成していることを表す．
[*2] ”BiLingual Evaluation Understudy” の略であり，機械翻訳の評価指標として良く用いられる．

(縮小付き内積注意) の考え方から式 2.1 のように計算する．

$$\mathrm{Attention}(Q,K,V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.1}$$

Multi-Head Attention は，Self-Atttention を複数並べることで精度の向上を図る仕組みである．Query，Key，Value を低次元にしたものをヘッドの数だけ作り，それぞれのヘッドで Attention を計算したものを Concat (結合) し，元の次元数に戻すことでより精度の高い Attention が実現できる．この計算を式 2.2 に示す．

$$\begin{aligned}\mathrm{MultiHead}(Q,K,V) &= \mathrm{Concat}(\mathrm{head}_1,\ldots,\mathrm{head}_h)W^O \\ \textbf{where}\ \ \mathrm{head}_i &= \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)\end{aligned} \tag{2.2}$$

式 2.2 の h はヘッド数を表す．また, モデルの次元数を d_{model}, Ouery と Key の次元数を d_k，Value の次元数を d_v と表す．このとき，$d_k = d_v = d_{\mathrm{model}}/h$ が成り立つ．重みは，それぞれ $W_i^Q \in \mathbb{R}^{d_{\mathrm{model}}\times d_k}$，$W_i^K \in \mathbb{R}^{d_{\mathrm{model}}\times d_k}$，$W_i^V \in \mathbb{R}^{d_{\mathrm{model}}\times d_v}$，$W^O \in \mathbb{R}^{hd_v\times d_{\mathrm{model}}}$ となる．

Add&Norm は，残差接続とレイヤー正規化を表す．残差接続は，Attention 層や FFN 層への入力と出力を加算する方法である．これにより，深いニューラルネットワークであっても勾配の消失を防ぐことができる．Attention 層や FFN 層への入力を x，出力を $f(x)$，残差接続層の出力を $h(x)$ とすると，式 2.3 のように表せる．

$$h(x) = f(x) + x \tag{2.3}$$

レイヤー正規化は，各層からの出力を層単位で正規化する働きを持つ．ミニバッチ単位で正規化を行うことをバッチ正規化というが，この正規化手法では小さいバッチサイズを設定した時に学習が不安定になってしまうという欠点があった．レイヤー正規化はその欠点を改善し，小さいバッチサイズであっても安定した学習を行うことを可能にした正規化手法である．

Feed Forward Network (FFN) は，入力値に非線形変換を行うために使用される仕組みである．非線形変換を行うための活性化関数には，ReLU 関数が用いられる．この操作は，x を FFN への入力ベクトル，W_1，W_2 を重み，b_1，b_2 をバイアスとしたとき，次式 2.4 のように表すことができる．

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{2.4}$$

Masked Multi-Head Attention は，Multi-Head Attention に現在の入力よりも先のトークンを隠す仕組みである．これにより，Transformer が次の単語を予測する際に未来の情報を参照してしまうことを防ぐことができる．

2.1.2　BERT

BERT [6] は，”Bidirectional Encoder Representations from Transformers” の略であり，2018 年に Google より公開された深層学習モデルである．BERT の大きな特徴は，双方向の Transformer Encoder によって構成されていることである．具体的には，図 2.3 のような構造を取る．これにより，文の前後を考慮した文脈理解が可能になり，より高度な自然言語処理タスクを実現できるようになった．

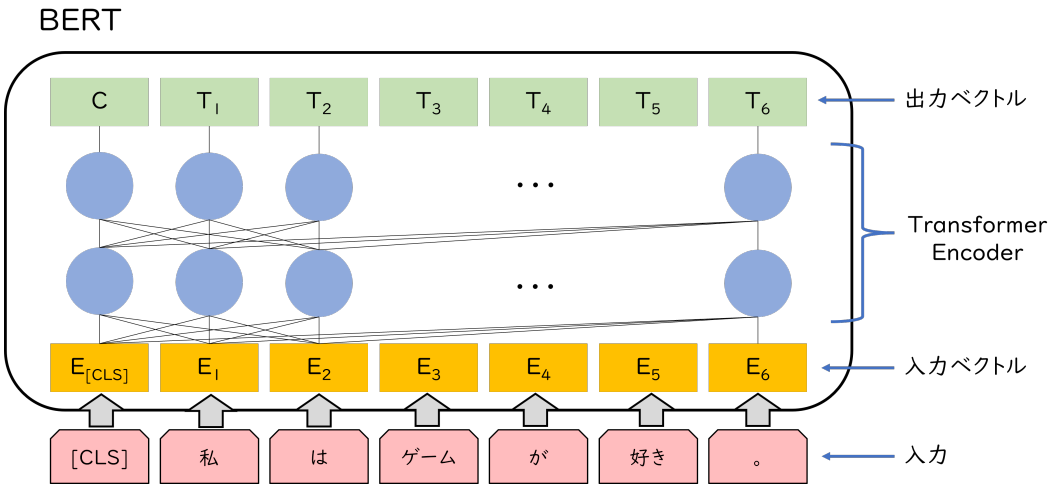

図 2.3: BERT の構造

事前学習

BERT は，大規模なテキストデータで事前学習されたモデルであり，Masked LM と Next Sentence Prediction(NSP) の 2 つの教師なしタスクにより学習を行っている．

Masked LM は，BERT が文の一部を特殊な単語に置き換えることで隠し，隠した単語を周りの単語から予測することで文脈を理解するタスクである．具体的には，文章全体からランダムに選ばれた 15% のトークンを”[MASK]” という特殊なトークンに置き換える．置き換えられた文章を BERT に入力し，”[MASK]” の位置に元々あったトークンを予測するというタスクを用いて学習を行う．

NSP は，2 つの文章の関係性を理解するために行うタスクである．具体的には，事前

学習に用いるテキストデータの中から 2 つの文章 A と B を選択する．このとき選ばれる文章 B の 50% は文章 A の後に続く文になっており，残りの 50% は後に続かない文章になっている．このように選ばれた文章 A と B が連続した文章になっているのかを予測することで 2 つの文章の関係性を学習する．

fine-tuning

fine-tuning とは，ラベル付きデータを用いて解きたいタスクに特化するようにモデルの学習を行う方法である．事前学習済みのモデルを利用するため，一から学習するよりも短時間で少ないデータ数で目的とするモデルを構築できるという利点がある．

BERT では，解きたいタスクに応じて新しい層を追加することでタスクに特化したモデルを作る．fine-tuning を行うとき，BERT は事前学習で得られたパラメータを初期値として用い，新たに追加された層ではランダムな値をパラメータとして用いる．そして，ラベル付きデータを用いることで BERT と追加した層の両方を学習することができる．

2.1.3　RoBERTa

RoBERTa [7] は，”Robustly optimized BERT approach” の略であり，2019 年に発表された深層学習モデルである．構成は，BERT モデルの仕組みをベースとしているが，事前学習時の設定にいくつかの調整を加えている．具体的には，以下の点で BERT と設定が異なっている．

- モデルに文が入力される毎にマスキングを行い，毎回違うトークンがマスクされた文章を学習できるようにする (Dynamic Masking).
- NSP による事前学習を廃止する．
- より長い文章を用いて事前学習を行う．
- 事前学習に使用するデータ量，バッチサイズ，学習回数を増加する．

RoBERTa は，当時高精度を記録していた BERT や XLNet を超える精度を記録し，多くの NLP タスクにおいて SOTA を達成した．

2.1.4　GPT-3 と GPT-3.5

GPT は，”Generative Pre-trained Transformer” の略であり，2018 年に OpenAI が提案した事前学習済みモデルである．構成は Transformer の Decoder をベースとしており，質問応答などのタスクで良好な結果を記録した．

GPT-3 [8] は 2020 年に公開された GPT の後継モデルの一つである．モデルの構成は GPT と大きな変化はないが，より大きなモデルに対してより多くのデータ量で事前学習を行っている点で異なる．パラメータは GPT が約 1 億 1700 万，GPT-2 は約 15 億なのに対して，GPT-3 は約 1750 億であり，GPT-2 から 100 倍以上に増加している．

BERT や GPT では，fine-tuning により事前学習済みモデルのパラメータを更新することで特定のタスクに特化したモデルを構築し，優れた性能を発揮していた．GPT-3 では，パラメータの更新を行わずに事前学習済みモデルにタスクを解かせるアプローチとして few-shot，one-shot，zero-shot の 3 つが紹介された．few-shot は推論時にタスクの説明と少量の具体例を与える方式である．同様に，one-shot は推論時にタスクの説明と一つの具体例を与える方式であり，zero-shot はタスクの説明のみを与え具体例はまったく与えない方式である．これらのアプローチは，GPT-3 に少量の具体例さえ与えれば様々なタスクに対応できるようになるため，fine-tuning のようにタスクごとのデータセットを構築する必要がないという点で優れている．図 2.4 には，few-shot, one-shot, zero-shot を実際に行う場合の具体例を示す．

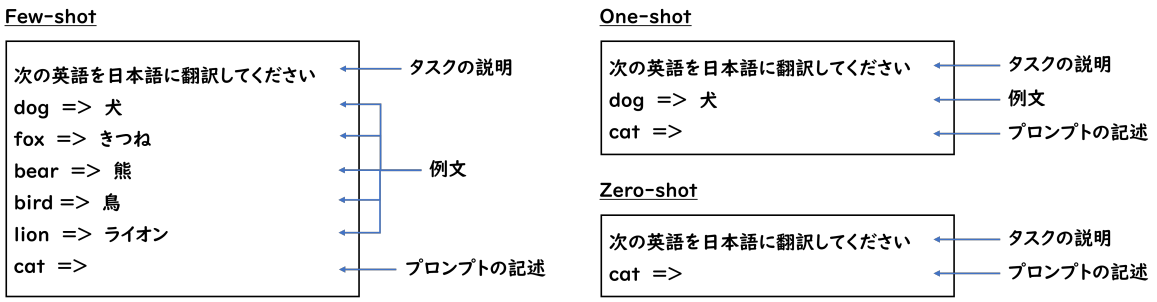

図 2.4: few-shot, one-shot, zero-shot の例

2022 年には，GPT-3 の後継モデルとなる GPT-3.5 が公開された．GPT-3.5 は詳細な情報が公開されていないが，GPT-3 よりも多くのパラメータ数を持つとされており，より精度の高い文生成が可能になっている．また，ChatGPT は GPT-3.5 に人間のフィードバックからの強化学習 (Reinforcement Learning from Human Feedback: RLHF) を

用いて人間が好む文を生成するように微調整したモデルがベースとなって作られている．

2.1.5　GPT-4

GPT-4 [3] は，2023 年に OpenAI が公開した GPT-3.5 の後継モデルである．パラメータ数などの詳細な情報に関しては GPT-3.5 と同様に公開されていないが，GPT-3.5 よりも遥かに創造的で信頼性が高く，より細かい指示に対応できるとされている．GPT-4 の大きな特徴には，RLHF の導入や画像の入力ができるようになったことが挙げられる．能力面では，人の受けるテストで好成績を記録し，多くの NLP タスクにおいて SOTA を達成するなど多くの成果を残している．

2.2　データ不足緩和に向けた研究

2.2.1　AnnoLLM

機械学習において，データにラベルを付与するプロセスのことをアノテーションという．NLP では，モデルを訓練して高い性能を得るためにラベル付きデータが必須である．しかし，アノテーション作業は人手で行われることが多く，必要となる時間とコストが高いことが問題となっている．

He ら [9] は，LLM にアノテーション作業を行わせる AnnoLLM という手法を提案した．この手法は，Chain-of-Thought (CoT) と呼ばれるタスクを解くまでの一連の手順を LLM に入力する文章 (プロンプト) に記述することで実現する．まず，ChatGPT (GPT-3.5) に解いてほしいタスクの説明と具体例，対応する正解ラベルを提示し，具体例に正解ラベルが対応する理由を説明させる．次に，アノテーションの方法と ChatGPT が生成した説明文を含む few-shot CoT プロンプトを作成する．そして，その few-shot CoT プロンプトを ChatGPT に入力することで LLM によるアノテーションを実現させる．図 2.5 は，この手順を示している．

AnnoLLM は，QK，BoolQ，WiC という 3 つの NLP タスクにおいて結果を残し，人間によるアノテーションの代替として LLM によるアノテーションが有効である可能性を示した．

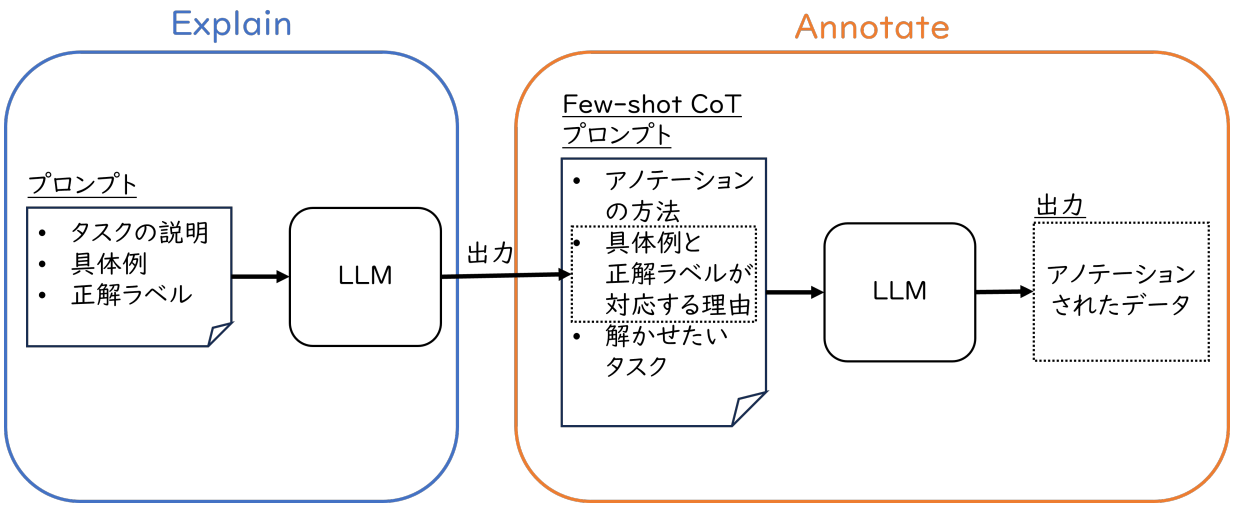

図 2.5: AnnoLLM の手順

2.2.2　Data Augmentation

少数の訓練データに対してデータを水増しすることでモデルの性能を向上させることを目的とする Data Augmentation というアプローチがある．これは，画像データやテキストデータといったデータセットに対応可能であり，画像処理や自然言語処理の分野で広く研究されている．

Li ら [10] は，自然言語処理における Data Augmentation のアプローチには 3 つのタイプがあることを示した．具体的には，表 2.1 に示した 3 タイプである

表 2.1: 自然言語処理における Data Augmentation

Paraphrasing	元データを少し言い換えることで元データとよく似たデータを生成する．
Noising	データの元々の意味が損なわれない程度にノイズを加えることで新たなデータを生成する．
Sampling	元データを利用して，まったく新しいデータを一から生成する．

Paraphrasing や Noising による Data Augmentation で有効性を示した手法の一つに EDA [11] という手法が存在する．EDA (Easy Data Augmentation) は，Wei らによって提案された言語モデルや外部データを必要としない簡単な Data Augmentation の手法である．具体的には，元データに対して表 2.2 に示した 4 つの方法で編集を行うことで新たなデータを生成する．この手法は，文書分類タスクにおいてモデルの性能を改善できることを示した．

表 2.2: EDA の例

元データ	I like to drive my car.
Synonym Replacement (類義語置換)	I like to drive my **vehicle**.
Random Insertion (ランダム挿入)	I like to drive my car **today**.
Random Swap (ランダム交換)	I **drive** to **like** my car.
Random Deletion (ランダム削除)	I like to drive car.

Sampling による Data Augmentation は，柔軟性の高いデータを生成できる手法である．しかし，他の 2 つのタイプに比べて，文生成に近いアプローチを取るため，難易度の高い手法でもある．そのため，高い文生成能力を持っている LLM を利用した Sampling による Data Augmentation の手法が広く検討されている．

Yoo ら [4] は，画像処理の分野に利用される Mix-up というアプローチを参考にし，GPT3Mix という手法を提案した．この手法では，文とラベルの組からなるプロンプトを作成し，GPT-3 に与えることで，新たな文とラベルの組を自動生成させる．図 2.6 は，感情分析における GPT3Mix の例を示している．GPT3Mix は，感情分析のようにラベルの種類が少ない文書分類タスクにおいて有効性があることを示した．

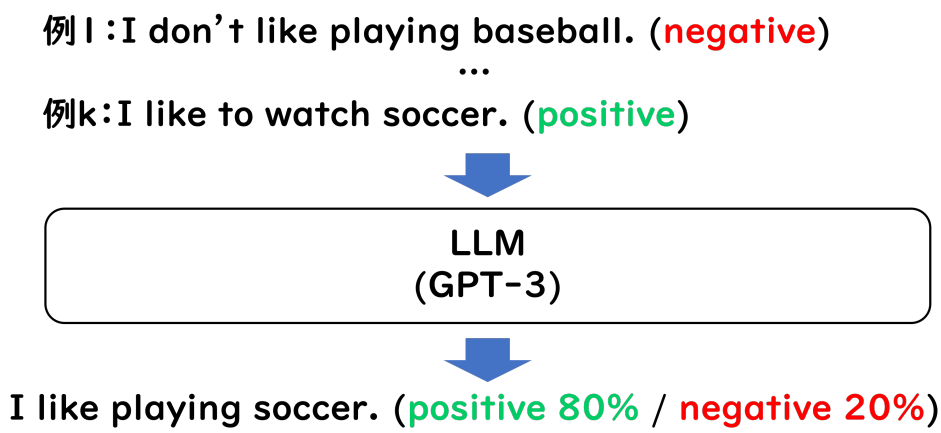

図 2.6: GPT3Mix によるデータ生成の例

Sahu ら [1] は特定の一つのラベルに属する文のみから構成される few-shot プロンプトを作成し GPT-3 に与えることで，ラベルを考慮した新たな文を生成する Data Augmentation の手法を提案した．図 2.7 は，この手法におけるデータ生成の具体例を示している．中町ら [2] は，特定の一つのラベルに属する文に加えて，キーワードとなる単語をプロンプトに与えることで，語彙と品質を考慮した文生成による Data Augmentation 手法を提案した．この実験では，日本語に特化した LLM である Hyper CLOVA を使用し，日本語データの Data Augmentation を試みている．図 2.8 は，この手法におけるデータ生成の手法を示している．

これらの Data Augmentation 手法において扱われるデータは全て短い文であり，長文のように few-shot プロンプトを作成するのが困難なデータに対する有効性は明らかになっていない．

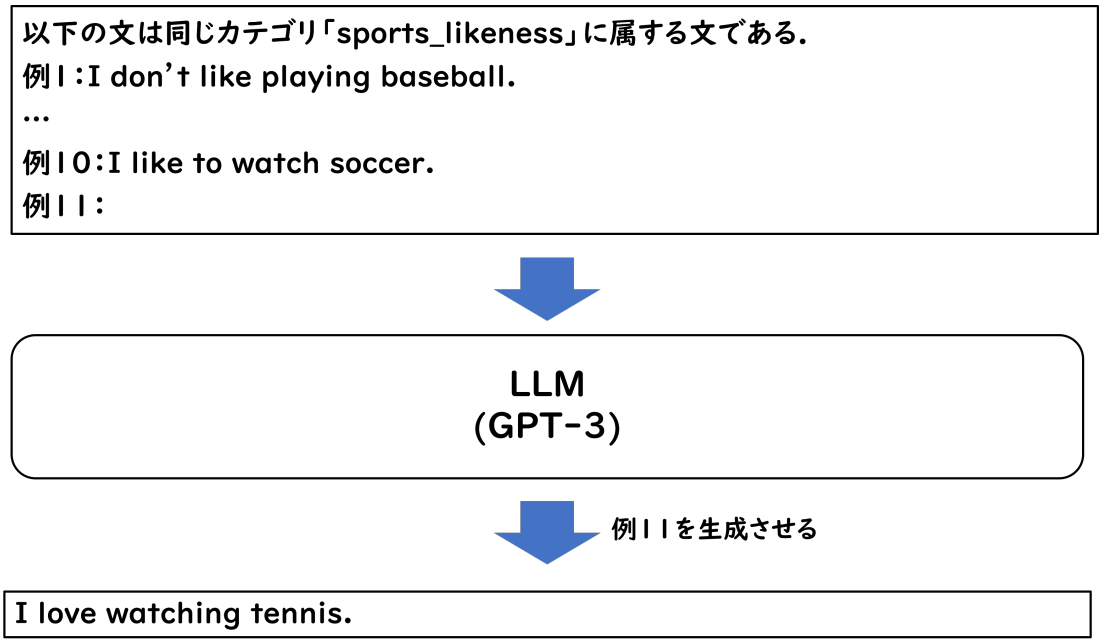

図 2.7: 論文 [1] で提案されたデータ生成手法の例

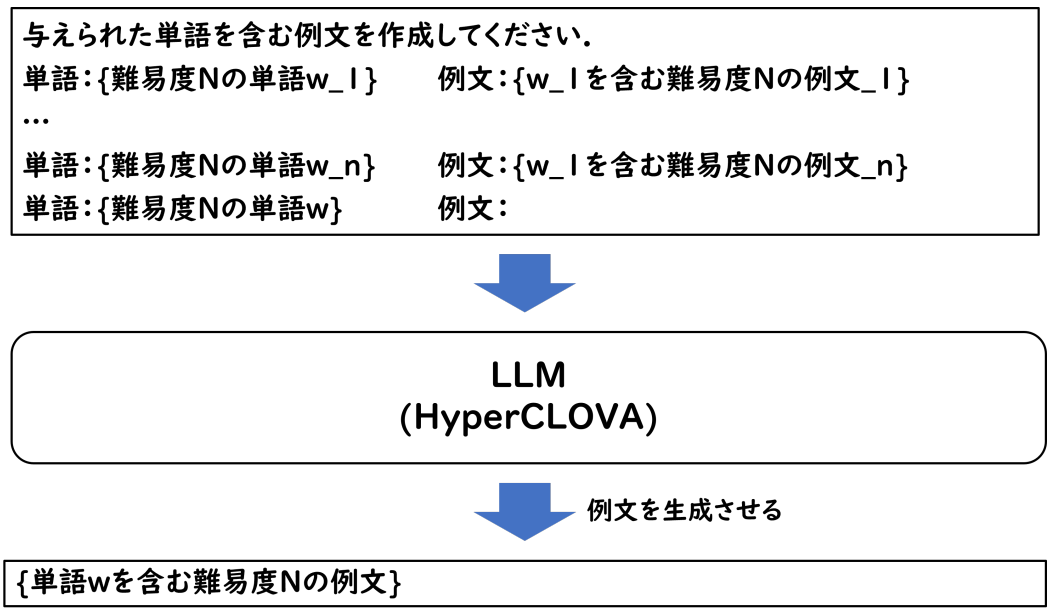

図 2.8: 論文 [2] で提案されたデータ生成手法

第 3 章

プロンプトの作成

GPT-3.5 や GPT-4 のような LLM は，タスクの詳細な説明や few-shot の例文をプロンプトに記述することでより良い文章生成を行うことができる．一方で，LLM が一度に処理できるトークン数は決まっており，一定のトークン数[*1]を超える長いプロンプトは処理ができない．そのため，多くの単語から成る長文データは few-shot による学習が困難である．本研究では，キーワードと要約文を使用することで文字数を抑えたプロンプトを作成し，長文データの自動構築を試みる．文生成を行うための LLM は，ChatGPT (GPT-3.5) を使用する．本研究の流れを図 3.1 に示す．

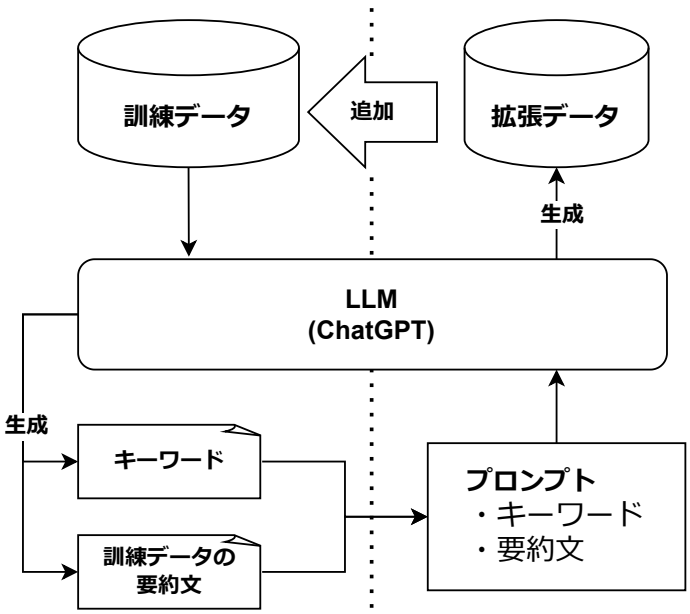

図 3.1: 本研究における Data Augmentation の流れ

[*1] https://platform.openai.com/docs/models 参照

3.1　プロンプトに使用するデータ

プロンプトに使用するデータとして，livedoor ニュースコーパス[*2]を使用する．これは，NHN Japan 株式会社が運営する「livedoor ニュース」からニュース記事を収集し，可能な限り HTML タグを取り除いたものである．このコーパスには，9 つのカテゴリのニュース記事が格納されている．

3.2　プロンプトの構成

LLM に与えるプロンプトには通常，入力トークン数の制限が存在する．そのため，数百から数千の文字数を持つ長文をそのまま例文として与えた few-shot プロンプトを使用するのは困難である．

本研究では，長文の要約文とそのキーワードから成るプロンプトを作成し，文字数を抑えた few-shot プロンプトによる長文データの拡張を試みた．図 3.2，図 3.3 及び図 3.4 は，本研究にて試みた 3 種類のプロンプトのテンプレートである．このように，キーワードや要約文の与え方を調整することで，生成される文に与える影響を比較した．

```
あなたは最高のニュース記事ライターです。
以下の制約を守ったニュース記事を書くことができます。例文は実際の記事の要約文の例です。
制約とキーワード、例文を参考にして、まったく新しいニュース記事を書いてください。

#制約
(1)生成する記事の文字数は1000文字程度とする
(2)記事全体の流れが自然になるようにする

#キーワード
{出力したいカテゴリのキーワード3つ}

#例文
例1:{出力したいカテゴリのテキストの要約文_1}
例2:{出力したいカテゴリのテキストの要約文_2}
…
例10:{出力したいカテゴリのテキストの要約文_10}

#出力
生成文1:
```

図 3.2: プロンプト 1 のテンプレート

[*2] https://www.rondhuit.com/download.html

```
あなたは最高のニュース記事ライターです。以下の制約を守ったニュース記事を書くことができます。
例文は9つにカテゴリ分けされたニュース記事の要約文を1つずつ提示したものです。
制約とキーワード、例文を参考にして、まったく新しいニュース記事を書いてください。

#制約
(1)生成する記事の文字数は1000文字程度とする
(2)記事全体の流れが自然になるようにする
(3)カテゴリ「{出力したいカテゴリ名}」の記事を書くこと

#キーワード
{出力したいカテゴリのキーワード3つ}

#例文
「独女通信」の例：{独女通信ラベルに属するテキストの要約文}
「ITライフハック」の例：{ITライフハックラベルに属するテキストの要約文}
…
「トピックニュース」の例：{トピックニュースラベルに属するテキストの要約文}

#出力
生成文1：
```

図 3.3: プロンプト 2 のテンプレート

```
あなたは最高のニュース記事ライターです。以下の制約を守ったニュース記事を書くことができます。
例文は実際のニュース記事の要約文です。
制約とキーワード、例文を参考にして、まったく新しいニュース記事を書いてください。

#制約
(1)生成する記事の文字数は1000文字程度とする
(2)記事全体の流れが自然になるようにする

#キーワード
{出力したいカテゴリのキーワード5つ}

#例文
例1：{出力したいカテゴリ以外のテキストの要約文_1}
例2：{出力したいカテゴリ以外のテキストの要約文_2}
…
例8：{出力したいカテゴリ以外のテキストの要約文_8}

#出力
生成文1：
```

図 3.4: プロンプト 3 のテンプレート

3.2.1　要約文の生成

要約文は，ChatGPT に元となるニュース記事の本文を一つずつ入力することで生成する (図 3.5 参照)．この際，ニュース記事の文体が損なわれるのを防ぐため，要約文を生成する際のプロンプトは，「以下の記事を文体を変えずに要約してください」という文に続く形でニュース記事本文を記述する．

本研究では，プロンプトごとに要約文の使用方法を変えている．各プロンプトにおける要約文の用い方については表 3.1 に示す．

表 3.1: プロンプトごとの要約文の使い方

プロンプト 1	特定の一つのカテゴリに属するニュース記事本文の要約文を 10 個与える．
プロンプト 2	各カテゴリに属するニュース記事本文の要約文を 1 個ずつ与える．
プロンプト 3	特定の一つのカテゴリに属するニュース記事本文の要約文を除く，他の 8 つのカテゴリのニュース記事本文の要約文を 1 個ずつ与える．

3.2.2　キーワードの生成

キーワードは ChatGPT に生成した要約文 10 個を与えることでカテゴリごとに生成させる (図 3.5 参照)．与えるキーワードの数は，プロンプトの種類により変わり，プロンプト 1 及びプロンプト 2 では 3 個，プロンプト 3 では 5 個とする．

CharGPT により生成されたキーワードは，以下の表 3.2 に示す．ここで下線の引いてある単語は，プロンプト 1 及びプロンプト 2 で使用される 3 個のキーワードを表す．

表 3.2: ChatGPT が生成したキーワード

カテゴリ名	キーワード
独女通信	ダイエット，女性，映画，女優，ヨーグルト
IT ライフハック	Ultrabook，Android，テクノロジー，アプリ，セキュリティ
家電チャンネル	スマートフォン，テクノロジー，プロジェクター，マッサージャー，メディア
Livedoor HOMME	借金，ファッション，ゴルフ，ドライブ，転職
MOVIE ENTER	映画，公開，日本，主題歌，俳優
Peachy	女性，ファッション，IPhone，イベント，チャリティ
SMAX	ソフトウェア更新，Android，スマートフォン，不具合修正，ＮＴＴドコモ
Sports Watch	W 杯，日本代表，選手，インタビュー，マッチ・試合
トピックニュース	ネット掲示板，ニュース記事，メディア，反応・コメント，イベント・出来事

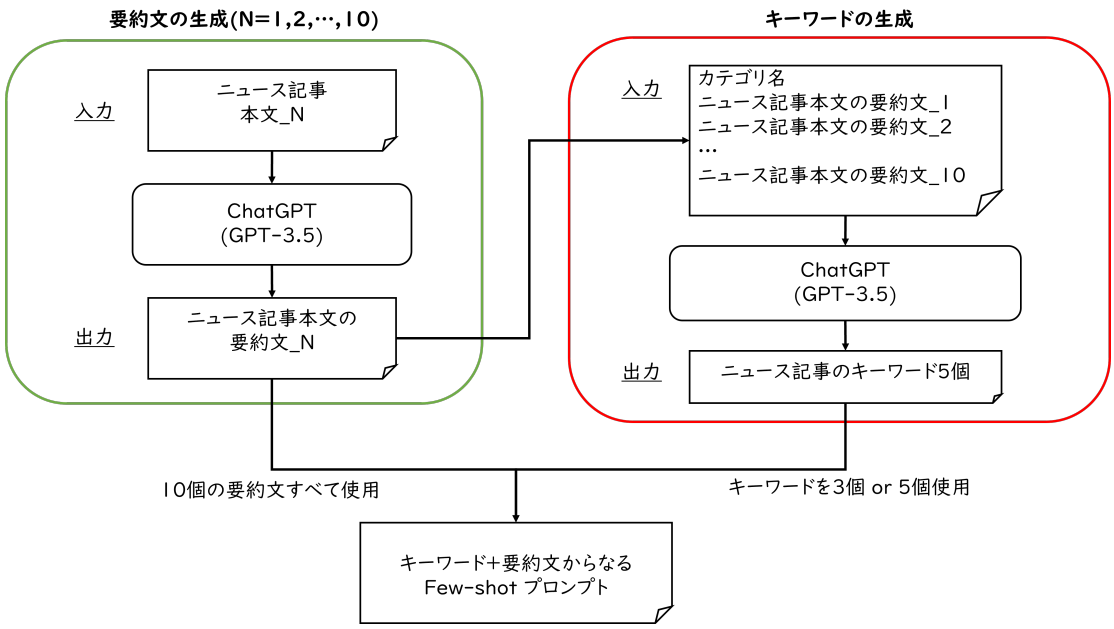

図 3.5: プロンプト作成までの流れ

第 4 章

実験

本研究における Data Augmentation の効果を評価するために livedoor ニュースコーパスを用いた文書分類タスクを行う．文書分類には事前学習済みモデルである RoBERTa を用い，fine-tuning を行うことで分類モデルを構築する．

4.1 文書分類に使用するモデル

rinna 社から公開されている日本語 RoBERTa モデルを使用した．これは Hugging Face 社の Transformers ライブラリから，”japanese-roberta-base”[*1]という名前で公開されているモデルである．このモデルは事前学習に日本語 Wikipedia と CC-100 が用いられており，パラメータ数は 111M である．またアーキテクチャは，Transformer Encoder が 12 層であり，隠れ層の次元数は 768 次元である．

4.2 データセット

本研究では，試みたプロンプトにより生成された拡張データを livedoor ニュースコーパスを用いた文書分類タスクで評価する．

データセットの構築では，まず 9 つのニュースカテゴリに対してラベルを割当て，各ラベルの文書をランダムに同数取り出す．そして，取り出したラベルとテキストの組から，訓練データ，検証データ，テストデータの 3 つを作成する．このときのデータの割当てを表 4.1 に示す．

[*1] https://huggingface.co/rinna/japanese-roberta-base

表 4.1: データセットの割当て

ラベル	カテゴリ名	訓練	検証	テスト
0	独女通信	10	10	100
1	IT ライフハック	10	10	100
2	家電チャンネル	10	10	100
3	Livedoor HOMME	10	10	100
4	MOVIE ENTER	10	10	100
5	Peachy	10	10	100
6	SMAX	10	10	100
7	Sports Watch	10	10	100
8	トピックニュース	10	10	100
	合計	90	90	900

表 4.1 に示した訓練データを元に拡張訓練データを作成する．拡張データは，提案するプロンプト一つにつきカテゴリごとに 3 件ずつ，すなわち 27 件のデータを作成する．したがって，拡張後の訓練データの数は元の訓練データ 90 件に拡張データ 27 件を加えた 117 件のデータとなる (図 4.1 参照).

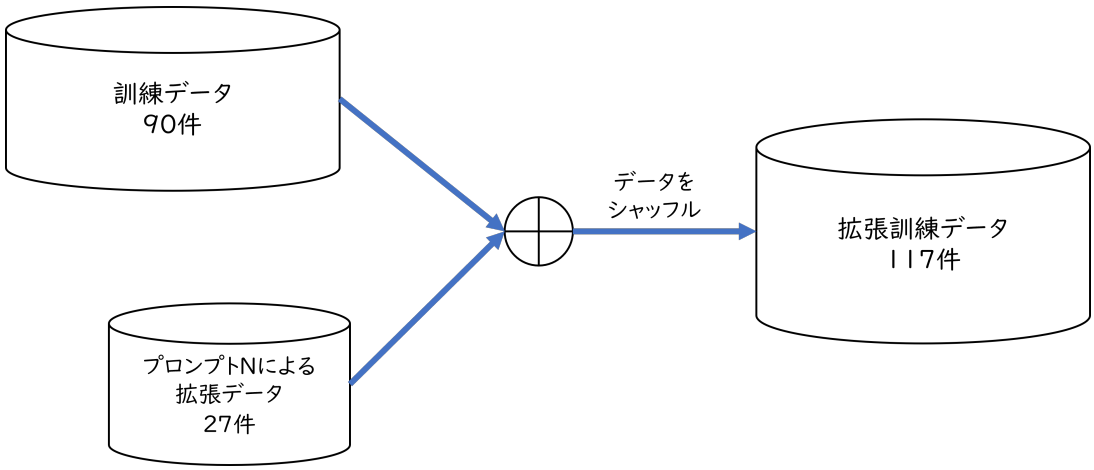

図 4.1: 拡張訓練データの作成

またデータの前処理として，訓練データとなるニュース記事からタイトルは除いている．つまり，本実験で取り扱うのはニュース記事本文のみとなるようにする．加えて，本

文に含まれるスペースやタブ文字，改行も取り除く．

4.3　文書分類モデルの学習

文書分類モデルの学習時の設定を以下に示す．

- 最適化関数：確率的勾配降下法 (SGD)
- 学習率：0.001
- バッチサイズ：3

エポック数の決定には，アーリーストッピングを用いる．アーリーストッピングとは，学習が進んでこれ以上の精度の改善が見込めない地点で学習を止める方法である．この方法は，モデルの過学習を防ぐために良く用いられている．

本実験では，検証データの損失が 3 エポック連続で最小値に更新されなかった場合に学習を中断し，モデルの構築を行う．このとき，学習の中断がなかった場合の最大エポック数は 100 と設定する．

4.4　実験結果

実験では，以下の 5 つのパターンにおける訓練データを使用する．これらの訓練データにより構築されたモデルによってテストデータの分類を行い，その正解率を出力した．この正解率を比較することで，試みたプロンプトによる Data Augmentation の効果を評価する．

元の訓練データ 90 件 (baseline)　各ラベル 10 件ずつランダムに抽出
拡張訓練データ (プロンプト 1)　baseline + プロンプト 1 による拡張データ 27 件
拡張訓練データ (プロンプト 2)　baseline + プロンプト 2 による拡張データ 27 件
拡張訓練データ (プロンプト 3)　baseline + プロンプト 3 による拡張データ 27 件
元の訓練データ 117 件　各ラベル 13 件ずつランダムに抽出

実験は訓練データ毎に 5 回ずつ行い，その平均正解率を記録した．この評価結果を表 4.2 及び図 4.2 に示す．評価結果より，元の訓練データ 90 件の平均正解率と比較すると，拡張訓練データ (プロンプト 1) は 0.0019，拡張訓練データ (プロンプト 2) は 0.0082，拡

張訓練データ (プロンプト 3) は 0.0082 低下した．一方で，元の訓練データ 90 件と元の訓練データ 117 件の平均正解率を比較すると，元の訓練データ 117 件の方が 0.0334 上昇していた．

表 4.2: モデルの評価結果

訓練データ	正解率
元の訓練データ 90 件	0.7655
拡張訓練データ (プロンプト 1)	0.7636
拡張訓練データ (プロンプト 2)	0.7573
拡張訓練データ (プロンプト 3)	0.7573
元の訓練データ 117 件	0.7989

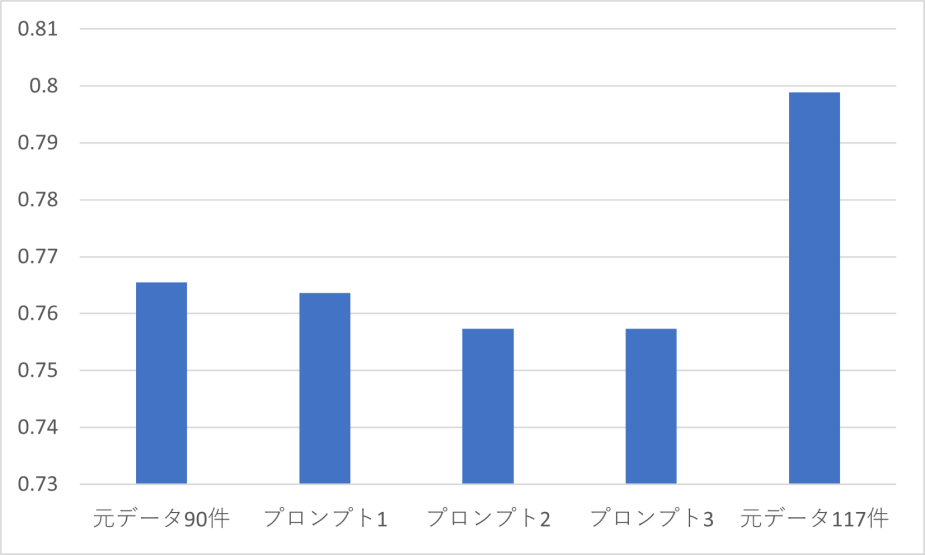

図 4.2: モデルの評価結果の比較図

第 5 章

考察

5.1 実験結果の考察

livedoor ニュースコーパスの文書分類において本プロンプトによる Data Augmentation を試みた場合，モデルの性能はわずかに低下してしまった．これは，LLM により生成した拡張データの中にノイズとなるデータが含まれていたことが要因だと考えられる．

本節では，実験におけるカテゴリごとの正解率を比較し，どのカテゴリにおいて LLM はノイズとなるデータを生成しやすいのかを確認した．以下の表 5.1 及び図 5.1 は，4 章の実験おけるカテゴリ毎の正解率を示す．

表 5.1: カテゴリごとの詳細な正解率

カテゴリ名	元データ 90	prompt1	prompt2	prompt3	元データ 117
独女通信	0.8266	0.8299	0.8433	0.8033	0.8332
IT ライフハック	0.5567	0.5732	0.5733	0.6267	0.6166
家電チャンネル	0.7332	0.7566	0.7467	0.73	0.74
Livedoor HOMME	0.6266	0.5933	0.5653	0.5599	0.6299
MOVIE ENTER	0.9133	0.9233	0.9166	0.9266	0.92
Peachy	0.5366	0.5567	0.5132	0.4999	0.69
SMAX	0.8299	0.7633	0.7699	0.7866	0.8566
Sports Watch	0.9333	0.9599	0.9633	0.9633	0.9666
トピックニュース	0.9333	0.9167	0.9233	0.92	0.9366

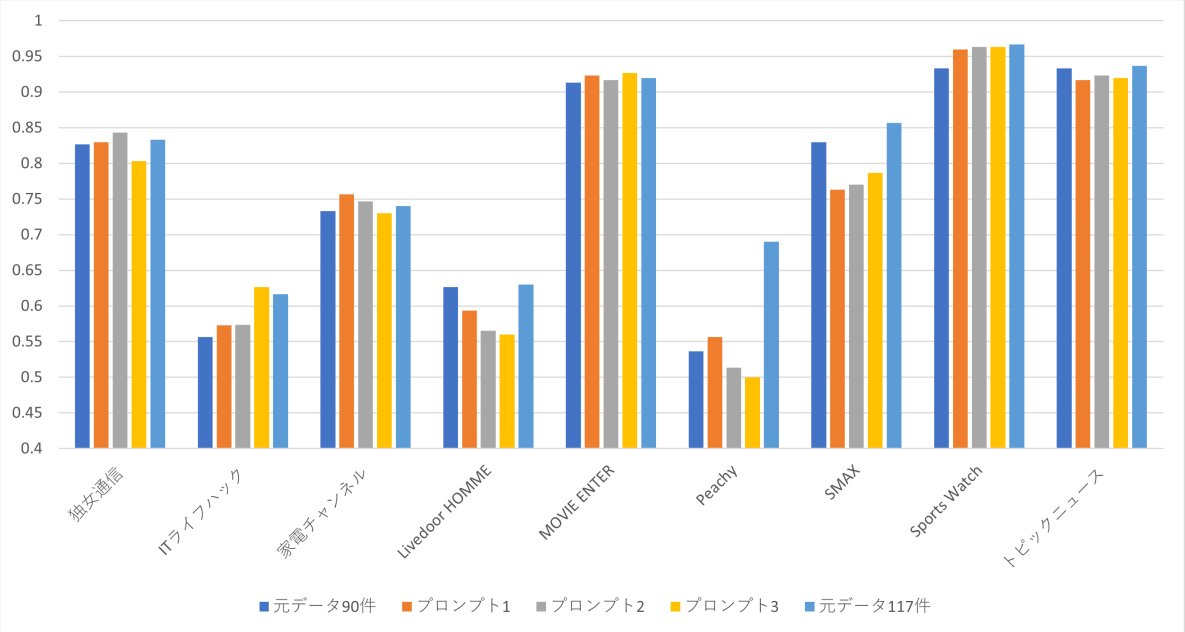

図 5.1: カテゴリごとの正解率の比較

Livedoor HOMME や SMAX，トピックニュースは，全てのプロンプトにおいて正解率が減少していることが読み取れる．これは，Livedoor HOMME やトピックニュースは多様な分野を取り扱うニュースカテゴリのため，LLM が少量の訓練データの中からニュース記事の特徴を学習するのが難しかったのではないかと考える．また，SMAX は似た内容のニュース記事を持つ IT ライフハックや家電チャンネルと分類モデルが混同してしまい正解率が減少してしまったと考えられる．

5.2　キーワード変更による実験

3.2 節では，ChatGPT にテキストの要約文を 10 個与えることでキーワードの生成を行っていた．しかし，ChatGPT のキーワードは 10 個の例を与えてもそのうちの 1 個の記事のキーワードが強く反映してしまうことがあり，全体的なキーワードとは言えないような単語もいくつか見られた．そのため，ChatGPT によるキーワードは多様性の高い文生成を抑制してしまう可能性がある．

本節では主観的ではあるが人手で要約文を読み，カテゴリごとのキーワードを独自で作成した．キーワードの作成する際は，固有名詞を使わないで各ニュースカテゴリを説明できるような単語を考える．これにより，与えたすべてのニュース記事の内容を踏まえたキーワードを作成した．本節でのプロンプトの作成の流れは，図 5.2 に示す．

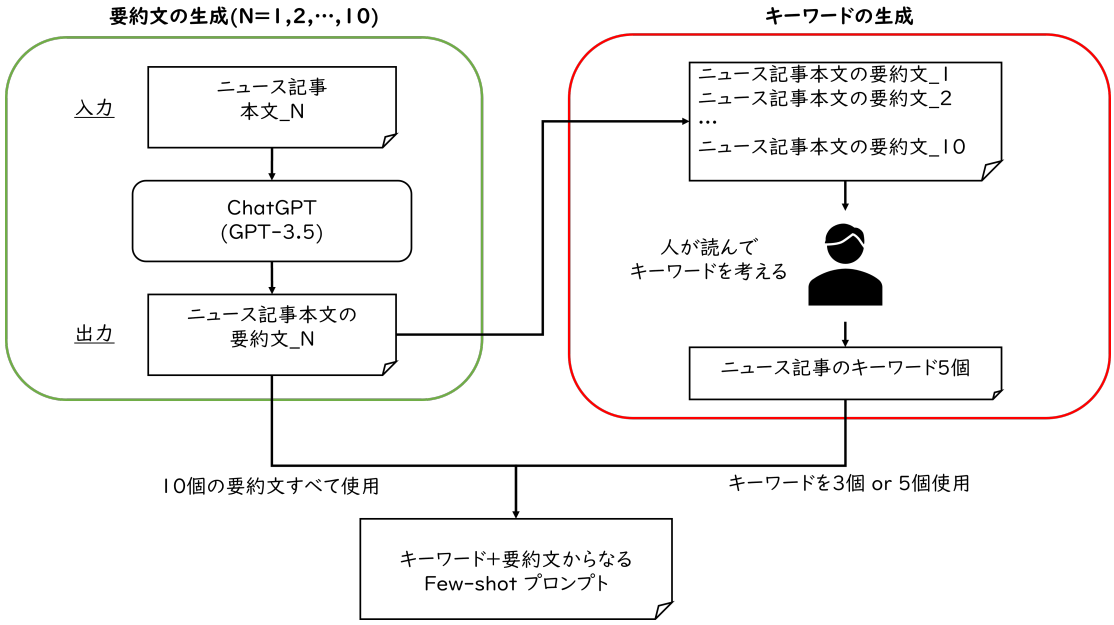

図 5.2: キーワード変更によるプロンプトの作成

ここで作成した具体的なキーワードは，表 5.2 に示す．プロンプト 1 とプロンプト 2 に与えるキーワードは表 5.2 の左から 3 個を使用した．

表 5.2: 人手で作成したキーワード

カテゴリ名	キーワード
独女通信	女性，恋愛，結婚，健康，美容
IT ライフハック	スマートフォン，タブレット，アプリ，セキュリティ，ネットワーク
家電チャンネル	家電，携帯電話，電子，機器，映像
Livedoor HOMME	仕事，ブランド，キャリア，転職，金
MOVIE ENTER	映画，主人公，監督，俳優，演技
Peachy	女性，ファッション，かわいい，手軽，イベント
SMAX	スマートフォン，モバイル，発表，機能，アップデート
Sports Watch	選手，試合，スポーツ，監督，記者
トピックニュース	芸能人，事件，政治，話題，意見

実験は，4 章と同様の設定で行った．表 5.3 及び図 5.3 は，その評価結果を示す．これを見ると，人手により作成したキーワードを使用したプロンプトによる拡張訓練データ

は元の訓練データ 90 件よりも平均正解率が減少していた．また，この実験の結果を元の訓練データ 90 件の結果と比較すると，平均正解率は同等かあるいはわずかに低下していた．これによりキーワードの変更は，本プロンプトにおいてモデルの性能を向上させることはできないことが分かった．すなわち、キーワードは本プロンプトにおいてモデルの性能に与える影響が小さいと考えられる．

表 5.3: キーワード変更時のモデルの評価結果

訓練データ	正解率
元の訓練データ 90 件	0.7655
拡張訓練データ (プロンプト 1)	0.7570
拡張訓練データ (プロンプト 2)	0.7552
拡張訓練データ (プロンプト 3)	0.76

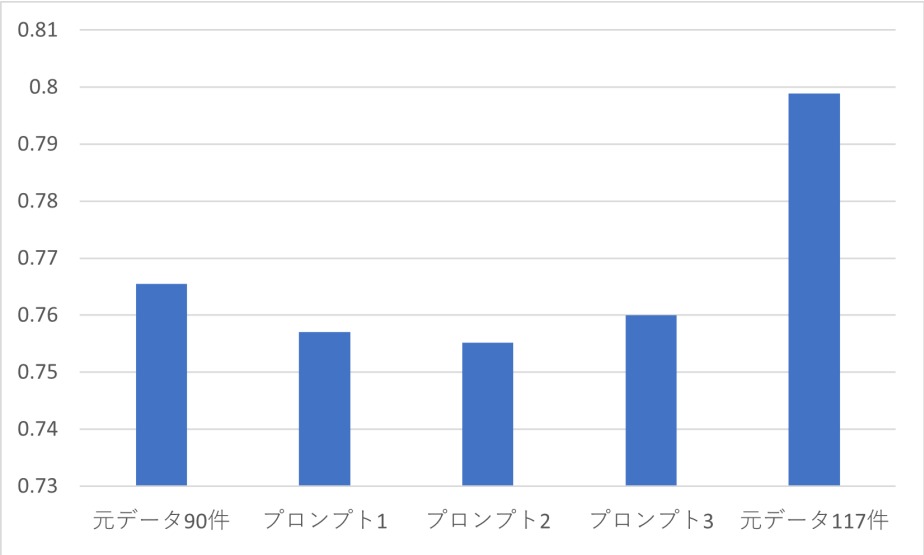

図 5.3: キーワード変更時のモデルの評価結果の比較図

5.3　データ数変更による実験

4.4 節では，各カテゴリ 3 件ずつ，合計 27 件の拡張による実験を行った．ここで有効性が見られなかった要因の一つとして，拡張するデータ数がカテゴリによっては少なすぎた可能性もある．本節では，拡張するデータ数を各カテゴリ 6 件ずつの合計 54 件とし，同様の実験を行い評価する．具体的には，以下のような訓練データを作成する．

元の訓練データ 90 件 (baseline)　各ラベル 10 件ずつランダムに抽出 (4.2 と同じもの)
拡張訓練データ (プロンプト 1)　baseline + プロンプト 1 による拡張データ 54 件
拡張訓練データ (プロンプト 2)　baseline + プロンプト 2 による拡張データ 54 件
拡張訓練データ (プロンプト 3)　baseline + プロンプト 3 による拡張データ 54 件
元の訓練データ 144 件　各ラベル 16 件ずつランダムに抽出

表 5.4 及び図 5.4 はこの評価結果を示す．これを元の訓練データ 90 件と比較すると，拡張訓練データ (プロンプト 1) は 0.0074，拡張訓練データ (プロンプト 2) は 0.118，拡張訓練データ (プロンプト 3) は 0.181 低下した．この結果より，本プロンプトによる拡張データを増やしていくと元の訓練データよりも正解率が低くなっていくことが分かった．これは，拡張データの数が増えたことによりノイズとなるデータが更に増えてしまったことや本プロンプトが ChatGPT において多くのデータを生成するのに適したものではなかったことが要因として考えられる．

表 5.4: データ数変更時のモデルの評価結果

訓練データ	正解率
元の訓練データ 90 件	0.7655
拡張訓練データ (プロンプト 1)	0.7581
拡張訓練データ (プロンプト 2)	0.7537
拡張訓練データ (プロンプト 3)	0.7474
元の訓練データ 144 件	0.7935

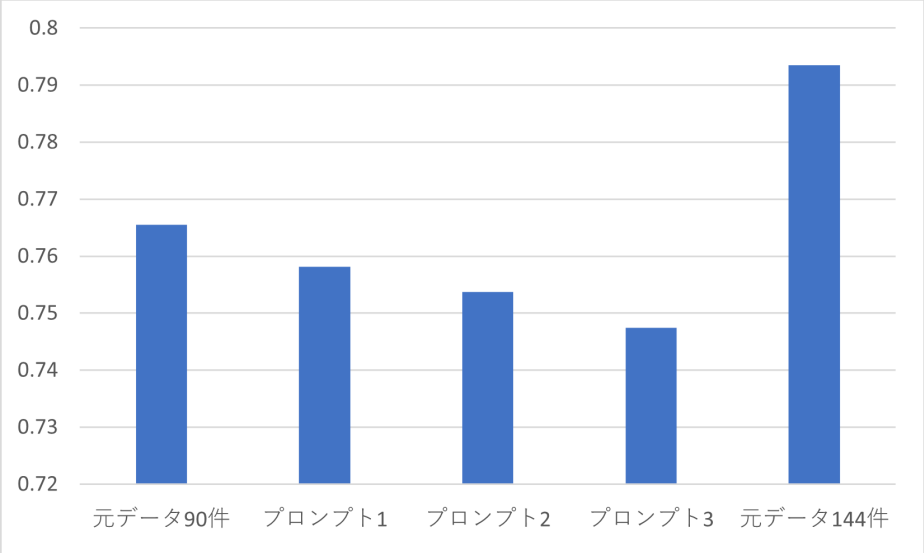

図 5.4: データ数変更時のモデルの評価結果の比較図

5.4　GPT-4 による生成文の試み

本研究では，ChatGPT の GPT-3.5 版を使用して拡張データを生成していた．本節では，より高い文章生成能力を持つ ChatGPT の GPT-4 による拡張データ生成を試みる．ここで，拡張データを生成するために GPT-4 に与えるプロンプトは 3.2 節と 5.2 節で作成した few-shot プロンプトとする．また，拡張データ数は各カテゴリ 1 件ずつとし，拡張訓練データは合計 99 件の訓練データとなる．この実験で使用する訓練データは以下に記す．

元の訓練データ 90 件 (baseline)　各ラベル 10 件ずつランダムに抽出 (4.2 節と同じもの)
拡張訓練データ (プロンプト 1)　baseline + プロンプト 1 による拡張データ 9 件
拡張訓練データ (プロンプト 2)　baseline + プロンプト 2 による拡張データ 9 件
拡張訓練データ (プロンプト 3)　baseline + プロンプト 3 による拡張データ 9 件

5.4.1　ChatGPT が生成したキーワードを使用したプロンプト

ChatGPT で生成したキーワードを使用した場合の few-shot プロンプトにおける評価結果を表 5.5 及び図 5.5 に示す．この結果を元の訓練データ 90 件と比較すると，平均正解率は拡張訓練データ (プロンプト 1) が 0.0054 上昇し，拡張訓練データ (プロンプト 2)

は 0.0011，拡張訓練データ (プロンプト 3) は 0.0035 低下していた．

表 5.5: ChatGPT 作成のキーワードにおける評価結果 (GPT-4)

訓練データ	正解率
元の訓練データ 90 件	0.7655
拡張訓練データ (プロンプト 1)	0.7709
拡張訓練データ (プロンプト 2)	0.7644
拡張訓練データ (プロンプト 3)	0.762

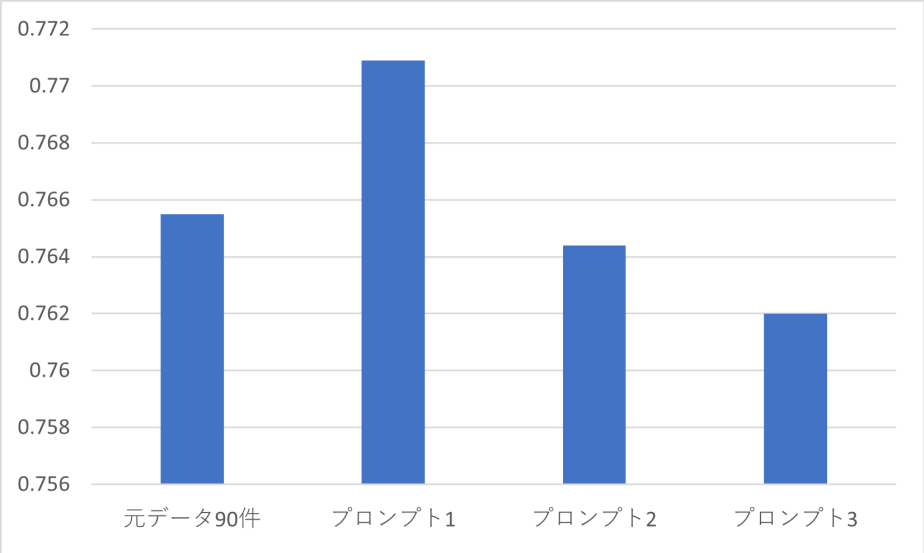

図 5.5: ChatGPT 作成のキーワードにおける評価結果の比較図 (GPT-4)

5.4.2　人手で作成したキーワードを使用したプロンプト

人手で作成したキーワードを使用した場合の few-shot プロンプトにおける評価結果を表 5.6 及び図 5.6 に示す．この結果を元の訓練データ 90 件と比較すると，平均正解率は拡張訓練データ (プロンプト 1) が 0.0035 減少し，拡張訓練データ (プロンプト 2) は 0.0055，拡張訓練データ (プロンプト 3) は 0.0018 上昇していた．

表 5.6: 人手作成のキーワードにおける評価結果 (GPT-4)

訓練データ	正解率
元の訓練データ 90 件	0.7655
拡張訓練データ (プロンプト 1)	0.762
拡張訓練データ (プロンプト 2)	0.771
拡張訓練データ (プロンプト 3)	0.7673

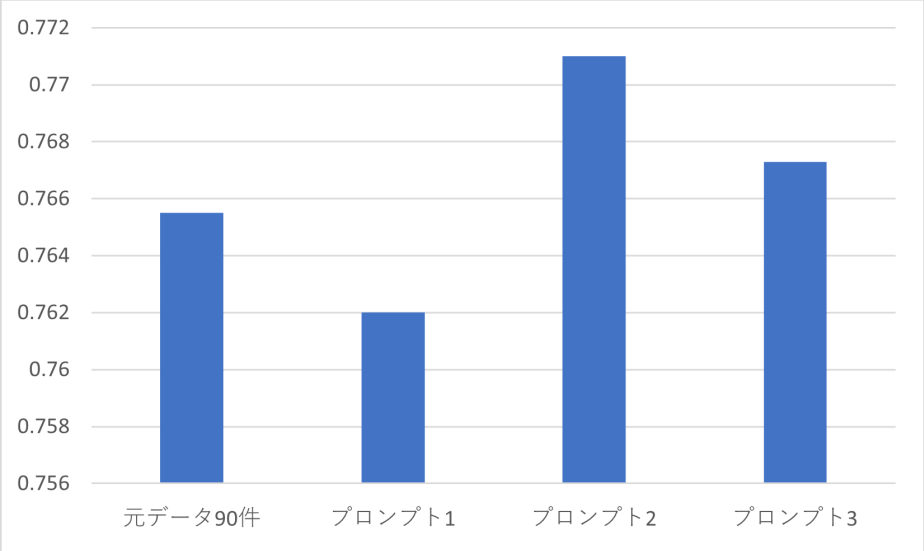

図 5.6: 人手作成のキーワードにおける評価結果の比較図 (GPT-4)

5.4.3　GPT-4 による生成文を使用した実験に関する考察

ChatGPT の GPT-4 版を用いて拡張データの生成を試みた場合，4 章で行った実験よりも拡張したデータ数が少ないにも関わらず最大 0.5% 程度精度が上昇するモデルがあった．この結果から，GPT-4 により生成された拡張データにより 4 章と同様のアプローチによる実験を行えば，分類モデルの精度が改善される可能性がある．

第 6 章

結論

本研究では，キーワードと文書の要約文により構成されたプロンプトによる LLM を利用した Data Augmentation の手法を試みた．キーワードや要約文の与え方を変えた 3 種類のプロンプトを考案し，日本語 RoBERTa による livedoor ニュースコーパスの文書分類タスクにより評価を行った．その結果，3 種類の拡張訓練データを使用した場合いずれもモデルの性能の向上は見られず，試みたプロンプトから生成された文による Data Augmentation の有効性を示すことはできなかった．

今後は，Data Augmentation のみに括らず，LLM による質の良いデータ生成とその活用方法の可能性の調査に取り組んでいきたい．

謝辞

本研究は国立国語研究所の共同研究プロジェクト「テキスト読み上げのための読みの曖昧性の分類と読み推定タスクのデータセットの構築」及び JSPS 科研費 23K11212 の助成を受けています．また本研究を進めるにあたって，新納浩幸教授には多くのご指導を頂きました．深く感謝いたします。加えて，新納研究室の皆様には日常の会話や議論を通して，多くの知識やご助言，ご協力を頂きました．厚く御礼申し上げます．

参考文献

[1] Gaurav Sahu, Pau Rodriguez, Issam Laradji, Parmida Atighehchian, David Vazquez, and Dzmitry Bahdanau. Data augmentation for intent classification with off-the-shelf large language models. In Bing Liu, Alexandros Papangelis, Stefan Ultes, Abhinav Rastogi, Yun-Nung Chen, Georgios Spithourakis, Elnaz Nouri, and Weiyan Shi, editors, *Proceedings of the 4th Workshop on NLP for Conversational AI*, pp. 47–57, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[2] 中町礼文, 西内沙恵, 浅原正幸, 佐藤敏紀. 語彙と品質を考慮したデータ水増しの言語教育支援への適用. 言語処理学会第 29 回年次大会発表論文集, pp. 1924–1929, 2023.

[3] OpenAI, :, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mo Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges,

Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon

Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2023.

[4] Kang Min Yoo, Dongju Park, Jaewook Kang, Sang-Woo Lee, and Woomyeong Park. Gpt3mix: Leveraging large-scale language models for text augmentation, 2021.

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

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[7] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[8] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners,

2020.

[9] Xingwei He, Zhenghao Lin, Yeyun Gong, A-Long Jin, Hang Zhang, Chen Lin, Jian Jiao, Siu Ming Yiu, Nan Duan, and Weizhu Chen. Annollm: Making large language models to be better crowdsourced annotators, 2023.

[10] Bohan Li, Yutai Hou, and Wanxiang Che. Data augmentation approaches in natural language processing: A survey. *AI Open*, Vol. 3, p. 71–90, 2022.

[11] Jason Wei and Kai Zou. Eda: Easy data augmentation techniques for boosting performance on text classification tasks, 2019.

付録

A　本研究で使用したプログラム

本研究で使用した主要なプログラムは以下の通りである.

A.1　Livedoor ニュースコーパスから訓練データとなる tsv ファイルを作成するコード

A.2　baseline の訓練データから拡張訓練データを作成するコード

A.3　テストデータにより文書分類を行うコード

ソースコード A.1: mktsvdata.py

```
#-*- coding:sjis -*-

import os
import glob
import numpy as np
import pandas as pd

# ------

def extract_main_txt(file_name):
    with open(file_name) as text_file:
        text = text_file.readlines()[3:]

        text = [sentence.strip() for sentence in text]
        text = list(filter(lambda line: line != '', text))
        text = ''.join(text)
        text = text.translate(str.maketrans(
            {'\n': '','\t': '','\r': '','\u3000': '','"': ''}))
        text = "[CLS]" + text
        return text
```

```
21
22  # -------
23
24  # files_folders = [name for name in os.listdir("./text/")]
25  # print(files_folders)
26
27  # カテゴリーフォルダのみを抽出
28  categories = [name for name in os.listdir("../text/") if os.path.
        isdir("../text/"+name)]
29
30  list_text = []
31  list_label = []
32
33  for cat in categories:
34      text_files = glob.glob(os.path.join("../text/",cat,"*.txt"))
35
36      # 前処理を実施して本文を取得
37      body = [extract_main_txt(text_file) for text_file in text_files]
38
39      label = [cat] * len(body) # カテゴリー名のラベルのリストを作成
40
41      list_text.extend(body) # リストごとに要素を追加する:extend
42      list_label.extend(label)
43
44  categories.sort()
45  print(categories)
46
47  df = pd.DataFrame({'text': list_text, 'label': list_label})
48
49  print(df.shape)
50
51  df.head()
52
53  # カテゴリーの辞書を作成
54  dic_id2cat = dict(zip(list(range(len(categories))), categories))
55  dic_cat2id = dict(zip(categories, list(range(len(categories)))))
56
57  print(dic_id2cat)
58  print(dic_cat2id)
59
60  # データフレームにカテゴリーインデックスの列を作成
```

```
61  df["label_index"] = df["label"].map(dic_cat2id)
62  df.head()
63
64  # ラベル列を消去し、テキスト，インデックスの順番にする
65  df = df.loc[:, ["label_index", "text"]]
66  df.head()
67
68  # カテゴリごとにデータフレームを作成
69  dokujo_df = df[df["label_index"]==0]
70  it_df = df[df["label_index"]==1]
71  kaden_df = df[df["label_index"]==2]
72  homme_df = df[df["label_index"]==3]
73  movie_df = df[df["label_index"]==4]
74  peachy_df = df[df["label_index"]==5]
75  smax_df = df[df["label_index"]==6]
76  sports_df = df[df["label_index"]==7]
77  topic_df = df[df["label_index"]==8]
78
79  # 訓練データ作成--------------------
80  train_df = pd.concat([dokujo_df.head(10),
81                        it_df.head(10),
82                        kaden_df.head(10),
83                        homme_df.head(10),
84                        movie_df.head(10),
85                        peachy_df.head(10),
86                        smax_df.head(10),
87                        sports_df.head(10),
88                        topic_df.head(10)], axis=0)
89
90  train_df = train_df.sample(frac=1, random_state=123).reset_index(
        drop=True)
91  train_df.to_csv("./dataset/train.tsv", sep='\t', index=False, header
        =None) #訓練データを
        保存
92
93  # 拡張------------------------------------------------------------
94
95  train_aug1_df = pd.concat([dokujo_df.head(11),
96                        it_df.head(11),
97                        kaden_df.head(11),
98                        homme_df.head(11),
```

```
                          movie_df.head(11),
                          peachy_df.head(11),
                          smax_df.head(11),
                          sports_df.head(11),
                          topic_df.head(11)], axis=0)

train_aug1_df = train_aug1_df.sample(frac=1, random_state=123).
    reset_index(drop=True)
train_aug1_df.to_csv("./dataset/train_aug.tsv", sep='\t', index=False
    , header=None)

# ------------------------------------------------------------

# 重複データ削除----------------------------------

df.drop(dokujo_df.index[:19], inplace=True)
df.drop(it_df.index[:19], inplace=True)
df.drop(kaden_df.index[:19], inplace=True)
df.drop(homme_df.index[:19], inplace=True)
df.drop(movie_df.index[:19], inplace=True)
df.drop(peachy_df.index[:19], inplace=True)
df.drop(smax_df.index[:19], inplace=True)
df.drop(sports_df.index[:19], inplace=True)
df.drop(topic_df.index[:19], inplace=True)

# 順番をシャッフル-----------------------------------------------

dokujo_df = dokujo_df.sample(frac=1, random_state=200).reset_index(
    drop=True)
it_df = it_df.sample(frac=1, random_state=200).reset_index(drop=True
    )
kaden_df = kaden_df.sample(frac=1, random_state=200).reset_index(
    drop=True)
homme_df = homme_df.sample(frac=1, random_state=200).reset_index(
    drop=True)
movie_df = movie_df.sample(frac=1, random_state=200).reset_index(
    drop=True)
peachy_df = peachy_df.sample(frac=1, random_state=200).reset_index(
    drop=True)
smax_df = smax_df.sample(frac=1, random_state=200).reset_index(drop=
    True)
```

```
131 sports_df = sports_df.sample(frac=1, random_state=200).reset_index(
        drop=True)
132 topic_df = topic_df.sample(frac=1, random_state=200).reset_index(
        drop=True)
133
134 # 検証データ作成--------------------------
135
136 val_df = pd.concat([dokujo_df.head(10),
137                     it_df.head(10),
138                     kaden_df.head(10),
139                     homme_df.head(10),
140                     movie_df.head(10),
141                     peachy_df.head(10),
142                     smax_df.head(10),
143                     sports_df.head(10),
144                     topic_df.head(10)], axis=0)
145
146 val_df = val_df.sample(frac=1, random_state=2022).reset_index(drop=
        True)
147 val_df.to_csv("./dataset/valid.tsv", sep='\t', index=False, header=
        None) #検証データ保
        存
148
149 # テストデータ作成------------------------
150
151 test_df = pd.concat([dokujo_df.tail(100),
152                     it_df.tail(100),
153                     kaden_df.tail(100),
154                     homme_df.tail(100),
155                     movie_df.tail(100),
156                     peachy_df.tail(100),
157                     smax_df.tail(100),
158                     sports_df.tail(100),
159                     topic_df.tail(100)], axis=0)
160
161 test_df = test_df.sample(frac=1, random_state=2023).reset_index(drop
        =True)
162 test_df.to_csv("./dataset/test.tsv", sep='\t', index=False, header=
        None) #テストデータ
        保存
```

ソースコード A.2: augment.py

```
1  # -*- coding: sjis -*-
2
3  import glob
4  import re
5  import random
6  import pandas as pd
7
8  # -------------------
9
10 def extract_text(text):
11     # 正規表現を使用してスペースやタブなどを取り除く
12     ex_text = re.sub(r'[\*\#\n\r\u3000\t\"]', '', text)
13     ex_text = "[CLS]" + ex_text # 文頭にトークンを付与[CLS]
14     return ex_text
15
16 def shuffle_tsv(file):
17     df = pd.read_csv(file, sep='\t')
18     shuffled_df = df.sample(frac=1, random_state=random.seed())
19     shuffled_df.to_csv(file, sep='\t', index=False)
20
21 # --------------------
22
23 dira = ["dokujo-tsushin", "it-life-hack", "kaden-channel", "livedoor
       -homme",
24         "movie-enter", "peachy", "smax", "sports-watch", "topic-news"
               ]
25
26 real_file = "./dataset/train.tsv" #訓練データのファイルを指定
27 filename1 = "./dataset/original/p1_aug.tsv" #拡張訓練データの保存先を
       指定
28 filename2 = "./dataset/original/p2_aug.tsv"
29 filename3 = "./dataset/original/p3_aug.tsv"
30
31 d1, d2, d3 = [], [], []
32
33 # プロンプト 1
34 for cls, dir in enumerate(dira):
35     files = glob.glob("./gpt-article/"+ dir + "/prompt1/original/*")
            #拡張データを取り
          出す
```

```
36      for file in files:
37          with open(file,'r',encoding='utf-8') as f1:
38              data = f1.read().split('\n')
39              text = str(cls) + "\t"
40              for i in range(len(data)):
41                  ex_data = extract_text(data[i])
42                  text += ex_data
43              d1.append(text)
44
45  with open(real_file,'r',encoding='utf-8') as ipt ,open(filename1,'w'
        ,encoding='utf-8') as out:
46      for line in ipt:
47          out.write(line)
48      for j in range(len(d1)):
49          out.write(d1[j])
50          out.write('\n')
51  shuffle_tsv(filename1)
52
53  # プロンプト 2
54  for cls, dir in enumerate(dira):
55      files = glob.glob("./gpt-article/"+ dir + "/prompt2/original/*")
56      for file in files:
57          with open(file,'r',encoding='utf-8') as f1:
58              data = f1.read().split('\n')
59              text = str(cls) + "\t"
60              for i in range(len(data)):
61                  ex_data = extract_text(data[i])
62                  text += ex_data
63              d2.append(text)
64
65  with open(real_file,'r',encoding='utf-8') as ipt ,open(filename2,'w'
        ,encoding='utf-8') as out:
66      for line in ipt:
67          out.write(line)
68      for j in range(len(d2)):
69          out.write(d2[j])
70          out.write('\n')
71  shuffle_tsv(filename2)
72
73  # プロンプト 3
74  for cls, dir in enumerate(dira):
```

```
75      files = glob.glob("./gpt-article/"+ dir + "/prompt3/original/*")
76      for file in files:
77          with open(file,'r',encoding='utf-8') as f1:
78              data = f1.read().split('\n')
79              text = str(cls) + "\t"
80              for i in range(len(data)):
81                  ex_data = extract_text(data[i])
82                  text += ex_data
83              d3.append(text)
84
85  with open(real_file,'r',encoding='utf-8') as ipt ,open(filename3,'w'
        ,encoding='utf-8') as out:
86      for line in ipt:
87          out.write(line)
88      for j in range(len(d3)):
89          out.write(d3[j])
90          out.write('\n')
91  shuffle_tsv(filename3)
```

ソースコード A.3: roberta_doccls.py

```
1   #-*-coding:sjis-*-
2
3   import os
4   import re
5   import torch
6   import pickle
7   import numpy as np
8   import pandas as pd
9   import pytorch_lightning as pl
10  import torch.nn as nn
11  import torch.optim as optim
12  import torch.nn.functional as F
13  from torch.utils.data import DataLoader, Dataset
14  from transformers import AutoTokenizer,
        AutoModelForSequenceClassification
15  from pytorch_lightning.callbacks import ModelCheckpoint,
        EarlyStopping
16  import warnings
17
18  # --- 警告を一部見えなくする設定---
```

```
19  warnings.simplefilter('ignore')
20  os.environ["TOKENIZERS_PARALLELISM"] = "false"
21  # --- グローバル変数の定義-------
22  NUM_LABELS = 9
23  MAX_EPOCH = 100
24  BATCH_SIZE = 3
25  TOKEN_LEN = 510
26  LABEL_COLUMN = "label"
27  TEXT_COLUMN = "text"
28  MODEL_NAME = "rinna/japanese-roberta-base"
29  # ------------------------------
30  file_path = "./dataset/"
31  file_kind = input("Enter␣the␣file␣name␣:␣")
32  # === データセットを生成するクラス===
33  class MakeDataset(Dataset):
34
35      def __init__(self, data, tknz, max_token_len):
36          self.data = data
37          self.tknz = tknz
38          self.max_token_len = max_token_len
39
40      def __len__(self):
41          return len(self.data)
42
43      def __getitem__(self, index):
44          data_row = self.data.iloc[index]
45          labels = data_row[LABEL_COLUMN]
46          text = data_row[TEXT_COLUMN]
47
48          encoding = self.tknz.encode_plus(
49              text,
50              add_special_tokens = True,
51              max_length = self.max_token_len,
52              padding = "max_length",
53              truncation = True,
54              return_attention_mask = True,
55              return_token_type_ids = True,
56              return_tensors = 'pt'
57          )
58
59          return dict(
```

```
60              input_ids = encoding["input_ids"].flatten(),
61              attention_mask = encoding["attention_mask"].flatten(),
62              token_type_ids = encoding["token_type_ids"].flatten(),
63              labels = torch.tensor(labels)
64          )
65  # ================================
66
67  # === データローダーを生成するクラス===
68  class MakeDataModule(pl.LightningDataModule):
69      def __init__(self, train_df, valid_df, test_df,
70                   batch_size = BATCH_SIZE, max_token_len = TOKEN_LEN,
71                   pretrained_model = MODEL_NAME):
72          super().__init__()
73          self.train_df = train_df
74          self.valid_df = valid_df
75          self.test_df = test_df
76          self.batch_size = batch_size
77          self.max_token_len = max_token_len
78          self.tknz = AutoTokenizer.from_pretrained(pretrained_model)
79
80      def setup(self):
81          self.train_dataset = MakeDataset(self.train_df, self.tknz,
                self.max_token_len)
82          self.valid_dataset = MakeDataset(self.valid_df, self.tknz,
                self.max_token_len)
83          self.test_dataset = MakeDataset(self.test_df, self.tknz,
                self.max_token_len)
84
85      def train_dataloader(self):
86          return DataLoader(self.train_dataset, batch_size=self.
                batch_size, shuffle=True, num_workers=os.cpu_count())
87
88      def val_dataloader(self):
89          return DataLoader(self.valid_dataset, batch_size=self.
                batch_size, num_workers=os.cpu_count())
90
91      def test_dataloader(self):
92          return DataLoader(self.test_dataset, batch_size=self.
                batch_size, num_workers=os.cpu_count())
93
94  # ================================
```

```
95
96  # === モデリングを行うクラス===
97  class MyClassificationModel(pl.LightningModule):
98
99      # --- モデルの構築とハイパーパラメータの保存---
100     def __init__(self, model_name, num_labels, lr):
101         super().__init__()
102         self.save_hyperparameters()
103         self.classification_model =
                AutoModelForSequenceClassification.from_pretrained(
104             model_name,
105             num_labels = num_labels
106         )
107
108     # --- 順伝搬---
109     def forward(self, batch):
110         output = self.classification_model(**batch)
111         return output
112
113     # --- 訓練---
114     def training_step(self, batch, batch_idx):
115         output = self.classification_model(**batch)
116         loss = output.loss
117         self.log('train_loss', loss)
118         return loss
119
120     # --- 検証---
121     def validation_step(self, batch, batch_idx):
122         output = self.classification_model(**batch)
123         val_loss = output.loss
124         self.log('val_loss', val_loss)
125
126     # --- テスト---
127     def test_step(self, batch, batch_idx):
128         labels = batch.pop('labels')
129         output = self.classification_model(**batch)
130         labels_predicted = output.logits.argmax(-1)
131         num_correct = (labels_predicted == labels).sum().item()
132         accuracy = num_correct / labels.size(0)
133         self.log('accuracy', accuracy)
134
```

```
135         # --- 最適化関数の設定---
136         def configure_optimizers(self):
137             return torch.optim.SGD(
138                 self.parameters(),
139                 lr = self.hparams.lr
140             )
141
142     # ================================
143
144     def my_collate_fn(batch):
145         images, targets = list(zip(*batch))
146         xs = list(images)
147         ys = list(targets)
148         return xs, ys
149
150     # --- チェックポイントの設定---
151     checkpoint = ModelCheckpoint(
152         monitor = 'val_loss', # 検証データの損失を監視
153         mode = 'min', # 検証データが最小の場合にチェックポイントを保存
154         save_top_k = 1, # 最も優れたモデルのみを保存
155         save_weights_only = True, # チェックポイントにモデルの重みのみを
                保存
156         dirpath = './models/',
157         filename = file_kind + '_{epoch}'
158     )
159
160     # --- アーリーストッピングの設定---
161     early_stopping = EarlyStopping(
162         monitor = 'val_loss', # 検証データの損失を監視
163         verbose = True, # 早期停止が起こった時にログメッセージを表示
164         mode = 'min' # 損失が増加したら訓練を停止する
165     )
166
167     # --- トレイナーの設定---
168     trainer = pl.Trainer(
169         accelerator = 'gpu', # 設定 GPU
170         max_epochs = MAX_EPOCH, # 訓練の最大エポック数
171         callbacks = [checkpoint, early_stopping]
172     )
173
174     # --- モデルインスタンスの作成---
```

```
model = MyClassificationModel(
    model_name = MODEL_NAME, # 事前学習済みモデル名
    num_labels = NUM_LABELS, # ラベル数
    lr = 0.001 # 学習率(Learning Rate)
)
print(model.hparams)

# ------------------------------------------------------------------------------


train_df = pd.read_csv(file_path + file_kind + ".tsv", sep="\t",
    names=["label", "text"])
valid_df = pd.read_csv(file_path + "valid.tsv", sep="\t", names=["
    label", "text"])
test_df = pd.read_csv(file_path + "test.tsv", sep="\t", names=["
    label", "text"])

data_module = MakeDataModule(train_df, valid_df, test_df)
data_module.setup()

item = data_module.train_dataset[0]
print(item["input_ids"].shape)

batch = next(iter(data_module.train_dataloader()))
print(batch["input_ids"].shape)

trainer.fit(
    model,
    train_dataloaders = data_module.train_dataloader(),
    val_dataloaders = data_module.val_dataloader()
)

result = trainer.test(
    ckpt_path = checkpoint.best_model_path,
    dataloaders = data_module.test_dataloader()
)

print(result)
```

B 本研究で生成した要約文

本研究では，拡張データを生成するためのプロンプトを構成する要素として要約文を ChatGPT (GPT-3.5) により生成した．以下の表 B.1 から B.9 には，10 個生成した要約文の中から 1 個の例をカテゴリ毎に示す．

表 B.1: 「独女通信」の要約文の例

多くの人が様々なダイエットを試みても、その継続が難しいことを経験しているでしょう。今回、ヨガインストラクターの城所恵美先生が、「続けやすい」ダイエット方法として、家で映画を観ながら行えるヨガを紹介します。この方法には Xbox 360 で利用できる Zune ビデオが使われており、インターネット接続さえあれば簡単に多くの映画を楽しむことができます。Zune ビデオは声で操作が可能なボイス・コントロール機能も搭載されており、ヨガのポーズをとりながら映画を楽しむのに最適です。最初に紹介されるポーズは「ウォーリア」で、アクション映画を見ながら行うことで、力強く勇敢な気分になることができます。次に紹介される「弓」のポーズは難易度が高いものの、同様に勇気を感じるポーズです。また、「鳩」のポーズは恋愛映画鑑賞に適しており、心が開放され感情豊かになる助けになります。サスペンス・ホラー映画には「舟」のポーズがおすすめで、身体が不安定なバランスポーズをとることで、恐怖感と緊張感をより感じやすくなります。最後に、SF 映画に対して「飛行機」のポーズが紹介されており、キープが難しいため高いダイエット効果が期待されます。映画ヨガは、映画好きの人にとって楽しみながらダイエットを続ける方法として魅力的です。紹介された映画はすべて Zune ビデオで配信されているため、映画ヨガを試しながら観賞することができます。これは Xbox 360、Zune ビデオ、および Zune ファンに関連する情報です。

表 B.2: 「IT ライフハック」の要約文の例

「Ultrabooker.jp」は、「Ultrabook」に特化したウェブサイトで、HP の美しい Ultrabook である「HP ENVY14-3000 SPECTRE」のレビューアーを募集しています。この Ultrabook は、トップカバーやディスプレイ、パームレスト、イメージパッドなど、傷に強いガラスを採用したボディが特徴です。この魅力的な Ultrabook を試すチャンスが登録ユーザーに提供されています。「HP ENVY14-3000 SPECTRE」は、14 インチワイドディスプレイを搭載し、Intel Core i7-2677M（1.8GHz）を搭載したスタイリッシュでハイパフォーマンスな Ultrabook です。「Ultrabooker.jp」は、Ultrabook に関する情報を提供する専門サイトで、製品紹介だけでなく、関連製品やおすすめアイテムなども紹介しています。ユーザーが実際に使って感じたことや良かった点などをリアルな言葉で共有するため、メーカーにとって貴重な情報源となっています。レビューアーの募集期間は 2012 年 4 月 10 日から 2012 年 4 月 24 日までで、対象商品は「Ultrabook HP ENVY14-3000 SPECTRE」で、2 名のレビューアーを募集しています。詳細情報は「Ultrabooker.jp」のウェブサイトで確認できます。

表 B.3: 「家電チャンネル」の要約文の例

テレビ向けのビデオ・オン・デマンドサービス「もっと TV」が、民放キー局 5 社と電通によって提供され、4 月 2 日からサービスを開始しました。しかし、このサービスはあまり話題になっていないようです。もっと TV は各局の新番組見逃しサービスを中心に、5000 本以上のコンテンツを提供しており、価格は 1 本あたり 100 円から 400 円まで設定されています。ラインナップには日本テレビの「三毛猫ホームズの推理」「HUNTER × HUNTER」、テレビ朝日の「W の悲劇」「都市伝説の女」、TBS テレビの「ハンチョウ〜警視庁安積班〜」「パパドル！」、テレビ東京の「クローバー」「イナズマイレブン GO」、フジテレビの「鍵のかかった部屋」「未来日記」などが含まれています。しかし、ネット上では 1 本あたりの価格が高すぎるとの批判が多く、実際にリアルタイムで見逃した番組を視聴者がどれだけの金額で見たいと思うかに疑問が呈されています。また、対応機種がパナソニックのテレビ「ビエラ」やデジタルレコーダー「ディーガ」などに限られているため、普及に制約がある可能性も指摘されています。同時に、似た名前の放送局「NOTTV」も同様の問題に直面しており、手軽な利用が難しいことが話題にならない理由の一つとされています。

表 B.4: 「Livedoor HOMME」の要約文の例

日本では、2011 年度末の国債や借入金などを合計した国の借金が過去最大の 1024 兆 1047 億円に達する見通しが明らかにされました。この増加の原因は、東日本大震災復興策の財源として復興債を発行したことなどが挙げられます。借金は予測不能な自然災害などによって、どんな状況に人々が立たされるかを考えさせられるものです。借金について言及するなら、福本伸行の大人気コミックを原作にした藤原竜也主演の映画『カイジ 2〜人生奪回ゲーム〜』が公開されました。この作品は借金にまみれた主人公カイジが人生を賭けた一発勝負に挑むストーリーで、公開初日 2 日間で 26 万人を動員し、約 3.5 億円の興行収入を記録するなど好スタートを切っています。観客の鑑賞理由は前作が面白かったことや原作のファンであることが主な要因とされています。また、ケータイ専用放送局「BeeTV」では、『賭博黙示録 カイジ』の Bee マンガが配信され、声優の演出によって楽しみが増えています。同局で放送されているドラマ『目を閉じてギラギラ〜一発逆転!?借金人生〜』では、野球のドラフトを巡る不正な取引によって借金に苦しむ天才投手の物語が展開されており、借金にまつわるドラマの人気も高まっています。

表 B.5: 「MOVIE ENTER」の要約文の例

10 月 22 日から、広島を舞台にした新たな日本映画『サルベージ・マイス』が広島で先行公開されます。この映画は、オール広島で撮影され、広島の観光名所で撮られた全く新しい作品で、ももいろクローバー Z が歌う主題歌「BIONIC CHERRY」のテレビスポットが解禁されました。若手演技派女優・谷村美月が新ヒロイン「サルベージ・マイス」を演じ、怪盗姿で見せるアクションは驚くべきものです。また、15 歳で空手歴 10 年の美少女空手家、長野じゅりあが演じる女子高生・美緒のアクションシーンも印象的です。広島の美しい風景が映画の舞台として魅力的に描かれ、ももいろクローバー Z の新曲「BIONIC CHERRY」が素晴らしい音楽として映像に調和しています。この映画は、正義の怪盗“サルベージ・マイス”が広島を舞台に美術品を狙う窃盗グループと戦う痛快なエンタテイメント作品です。監督は特撮映画の経験が豊富な田崎竜太で、プロデューサー兼アクション監督は日本のリアルアクション映画の先駆者である西冬彦が担当しています。ももいろクローバー Z の主題歌「BIONIC CHERRY」は、11 月 23 日に発売される彼らのニューシングルに収録される予定です。『サルベージ・マイス』の公開情報はこちらで確認できます。

表 B.6: 「Peachy」の要約文の例

「iPhone 女子部」は、全国に 2 万 1000 人以上の部員を持つ組織で、女性向けの iPhone アクセサリーを集めた「ガールズ アクセサリー ショー vol.1」というイベントを開催します。この組織のコンセプトは、「iPhone と一緒の楽しい女子ライフ」で、女性向けのオシャレな iPhone ライフを提案しています。彼女らは Twitter で交流し、デコのワークショップを行ったり、部員だけの情報交換会で可愛いアイテム、楽しいこと、新しいトレンドを発信しています。このイベントでは、女性が好きな iPhone アクセサリーが一堂に集結します。限定の iPhone アクセサリーの販売や、「Wool cube wool」というコラージュ作家によるデコケース作りワークショップなども行われます。また、日本最大のアプリレビューサイト「AppBank」や iPhone 界の有名人を招いたトークショーも予定されています。占いブースやフードユニット「カリ〜番長」の女子カリ〜販売、女性向けのプレゼント企画など、さまざまな企画が用意されています。このイベントは、iPhone をもっと可愛く楽しむための女性にぴったりの機会です。開催日時は、3 月 9 日から 3 月 11 日までで、場所は東京の BA-TSU ART GALLERY です。 iPhone 愛好者にとって、見逃せない楽しいイベントとなりそうです。

表 B.7: 「SMAX」の要約文の例

NTT ドコモは Android スマートフォン「AQUOS PHONE f SH-13C」と「Q-pot.Phone SH-04D」における Web ブラウザの不具合を修正するソフトウェア更新を提供開始しました。この更新では、以下の 2 つの主要な不具合が修正されます。まず、特定のサイトでファイルをアップロードすると、完了後の画面で一部文字化けが発生する問題があり、また、Google Map サイトでドラッグ操作を行う際に地図表示がチラつく場合も修正されます。更新は自動でダウンロードされ、あらかじめ設定した時間に書き換えが行われます。また、ユーザーが手動で即時更新も行うことができます。ただし、ソフトウェア更新を行う際には注意が必要で、バックアップを取ることや電池をフル充電することが推奨されています。また、一部の条件下ではソフトウェア更新ができない場合があります。この更新は、ユーザーが快適にスマートフォンを利用できるようにするためのものであり、アプリケーションの使用に影響を及ぼす可能性もあるため、注意が必要です。更新に失敗した場合は、故障取扱窓口に連絡する必要があります。なお、ユーザーの情報はソフトウェア更新以外の目的には利用されないとのことです。

表 B.8: 「Sports Watch」の要約文の例

田中将大（23)、東北楽天のエースとタレントの里田まい（27）が結婚を発表しました。田中は自身のブログで、彼女のサポートが昨シーズンの成功に貢献したことを述べ、今後の活躍に期待を寄せました。一方、元監督の野村克也氏は、結婚の知らせを新聞記者から聞いて驚き、田中が挨拶に来なかったことに不満を表明しました。野村氏は野球選手が社会に対して無知であると指摘し、年上の女性との結婚を推奨し、祝福の言葉を述べました。

表 B.9: 「トピックニュース」の要約文の例

8 月 26 日、橋下氏が文楽協会への補助金凍結の可能性を示唆し、その後に古典芸能を鑑賞した際の感想を述べたと、各新聞が報じており、これがネット掲示板で大きな反響を呼んでいます。橋下氏は 2009 年に古典芸能を初めて鑑賞した際、「二度と観に行かない」と酷評していましたが、今年 8 月 26 日に国立文楽劇場で文楽の古典「曽根崎心中」を観賞しました。その後、橋下氏は「古典として守るべき芸だと分かったが、新たなファンを獲得するためには台本が古すぎるため、現代風に演出をアレンジするなどの工夫が必要だ」と述べました。大阪市は、今年度の補正予算案で文楽協会への補助金を、前年度に比べて 25 %減少の 3900 万円としました。橋下氏は今月初め、「公開での面談でないと補助金を支出しない」と表明しましたが、協会側は非公開での面談を求めたため、対立状態が続いています。ネット掲示板では、橋下氏に対する非難が相次ぎ、「なんでこんなバカっぽく見える発言するんだよ ww」「古典芸能を見て台本が古すぎるって意味分からん」「歌舞伎があるから文楽なんていらねえ、儲からないものはいらねえって思ってるんだろうな」とのコメントが寄せられました。中には、不倫騒動を引き合いにした皮肉交じりのコメントもありました。27 日、この問題について橋下氏は自身のツイッターで釈明を行っています。

C 本実験で生成された拡張データ

本論文の 4 章で行った実験で生成された拡張データの例を表 C.10 から C.12 に示す．ただし，本節では「独女通信」カテゴリに属するような文書を ChatGPT (GPT-3.5) に生成させた場合の例を 1 個記すとする．

表 C.10: プロンプト 1 の生成例

多くの女性が悩む「ダイエット」。その継続が難しいことはよく知られています。しかし、今回、新たな方法が登場し、女性たちに大きな魅力を持って受け入れられています。ヨガインストラクターの城所恵美先生が、「続けやすい」ダイエット方法として、家で映画を観ながら行える「映画ヨガ」を紹介し、多くの女性たちの注目を浴びています。この方法の魅力は、家庭で手軽に実践できる点です。必要なのはインターネット接続と Xbox 360 で利用できる Zune ビデオのみ。このアプローチは、映画好きな女性にとって理想的で、ダイエットを楽しみながら続けることができます。「映画ヨガ」では、異なるヨガポーズがさまざまな映画ジャンルに合わせて紹介されています。たとえば、アクション映画を鑑賞する際には「ウォーリア」のポーズをとることで、力強く勇敢な気分を味わえます。恋愛映画には「鳩」のポーズが適しており、心が開放され感情豊かになる手助けになります。サスペンス・ホラー映画を楽しむ際には「舟」のポーズがおすすめで、身体が不安定なバランスポーズをとることで、恐怖感と緊張感をより感じやすくなります。そして、SF 映画に対して「飛行機」のポーズが紹介されており、キープが難しいため高いダイエット効果が期待されます。映画ヨガは、ダイエットだけでなく、映画鑑賞の楽しみも提供します。紹介された映画はすべて Zune ビデオで配信されており、映画ヨガを試しながら観賞することができます。これは Xbox 360、Zune ビデオ、および Zune ファンに関連する情報で、多くの女性たちにとって魅力的なダイエット方法として注目されています。映画ヨガを通じて、ダイエットを続けつつ映画の楽しみを味わう新たなアプローチが女性たちに提供されています。

表 C.11: プロンプト 2 の生成例

多くの女性が悩む「お腹の張り」について、新たな映画が注目を集めています。女性たちの心をつかむダイエット映画が、10 月 22 日から公開される予定です。この映画は、広島を舞台にした新たな日本映画で、若手演技派女優・谷村美月が新ヒロイン「サルベージ・マイス」を演じます。怪盗姿で活躍する美緒役を演じるのは、15 歳で空手歴 10 年の美少女空手家、長野じゅりあです。この映画は、正義の怪盗“サルベージ・マイス”が広島を舞台に美術品を狙う窃盗グループと戦う痛快なエンタテイメント作品で、監督は特撮映画の経験が豊富な田崎竜太が務め、プロデューサー兼アクション監督は日本のリアルアクション映画の先駆者である西冬彦が担当しています。さらに、ももいろクローバー Z が歌う主題歌「BIONIC CHERRY」が、映画の魅力を一層引き立てています。この映画は、広島の美しい風景を背景に、アクションとスリリングなストーリーが織り成す、ダイエットに励む女性たちに贈る新たな映画として注目されています。「サルベージ・マイス」は、10 月 22 日から広島で先行公開され、映画館での待ち時間にお腹の張りを気にする女性たちに、元気と刺激を届けることでしょう。新たな映画体験とともに、美緒のアクションに魅了され、自身も元気になりたいと考える女性たちにとって、この映画は必見です。ダイエットに悩む女性たちに贈る、笑顔と活力に満ちた映画が、今秋、映画館で待っています。お腹の張りを忘れ、美緒の活躍に心を奮い立たせて、新たなスタートを切りましょう。

表 C.12: プロンプト 3 の生成例

近年、女性たちの間でヨーグルトを活用したダイエット法が注目されています。ヨーグルトは健康的な食材として広く知られており、その中でもプロバイオティクスを含むヨーグルトが、ダイエットのサポートに効果的であると言われています。プロバイオティクスは腸内の健康を促進し、代謝を向上させる働きがあります。そのため、ダイエット中の女性たちにとって、ヨーグルトは味方として注目されているのです。さらに、女性向けの映画界でも注目すべき出来事が起きています。新たな女優、美緒が注目の作品に出演し、そのアクションシーンが話題となっています。美緒は 15 歳で空手歴 10 年の美少女空手家として知られ、その実力を活かした演技が観客に強い印象を与えています。映画『サルベージ・マイス』は、広島を舞台にした作品で、広島の美しい風景とももいろクローバー Z の新曲「BIONIC CHERRY」が相まって、観客を引き込んでいます。監督には特撮映画の経験が豊富な田崎竜太が、プロデューサー兼アクション監督には日本のリアルアクション映画の先駆者である西冬彦が担当しており、期待の高い作品となっています。一方、女性たちの日常での気分転換方法についてもアイシェアのアンケートから興味深い結果が明らかになりました。軽食やお茶、コーヒーを楽しむことが、女性たちのやる気を引き立てる方法として挙げられています。スターバックスの「スターバックス VIA コーヒーエッセンス」は、手軽に美味しいコーヒーを楽しむためのアイテムとして女性たちに支持されています。これらの要素が組み合わさり、女性たちの日常生活におけるダイエットやリラックス方法、エンターテインメントに新たな風を吹き込んでいます。女性たちの生活に彩りを添える情報に注目です。

D 人手作成キーワードに変更したプロンプトによる拡張データ

5.2 節において生成した，キーワード変更による拡張データから「独女通信」の一例を以下の表 D.13 から D.15 に示す.

表 D.13: 人手キーワードにおけるプロンプト 1 の生成例

近年、女性たちの恋愛における新しいトレンドが注目されています。それは「映画ヨガ」です。多くの人が様々なダイエットに挑戦しても、その継続が難しいと感じている中、ヨガインストラクターの城所恵美先生が提案するのは、家で映画を観ながら行うヨガです。この方法では、Xbox 360 で利用できる Zune ビデオを活用し、インターネット接続さえあれば手軽に多くの映画を楽しむことができます。Zune ビデオは声で操作が可能なボイス・コントロール機能も備えており、ヨガのポーズをとりながら映画を楽しむのに最適です。例えば、「ウォーリア」のポーズをアクション映画とともに行うことで、力強く勇敢な気分になることができます。また、「弓」のポーズは難易度が高いものの、同様に勇気を感じるポーズです。恋愛映画には「鳩」のポーズがぴったりで、心が開放され感情豊かになる助けになります。サスペンス・ホラー映画に挑む際は「舟」のポーズがおすすめで、身体が不安定なバランスポーズをとることで、恐怖感と緊張感をより感じやすくなります。最後に SF 映画に対して「飛行機」のポーズが紹介されており、キープが難しいため高いダイエット効果が期待されます。映画ヨガは、映画好きの女性たちにとって楽しみながらダイエットを続ける方法として魅力的です。紹介された映画はすべて Zune ビデオで配信されているため、映画ヨガを試しながら観賞することができます。これは Xbox 360、Zune ビデオ、および Zune ファンに関連する情報です。新しいトレンドの誕生で、女性たちの健康とエンターテインメントが融合する新たな可能性が広がりつつあります。

表 D.14: 人手キーワードにおけるプロンプト 2 の生成例

多くの女性が悩む「お腹の張り」に関する新たな調査結果が明らかになりました。トレンダーズが 25〜49 歳の女性モデル 50 名に行った調査によれば、お腹の張りに悩んでいると回答したモデルは 74 %に上り、87 %がお腹が張ることを恥ずかしいと感じていることがわかりました。この悩みは外見だけでなく、精神的な影響も及ぼすことがあり、撮影時に笑顔が作れないという声もあります。モデルたちはお腹の張りに対処するために、食事に工夫をしています。調査によれば、最も多くのモデルが摂取しているのは「ヨーグルト」で、46 %が愛用しています。他にも「野菜全般」（12 %）や「根菜全般」（12 %）、そして「玄米」（10 %）も食事に取り入れていることが分かりました。特に、プロバイオティクス系のヨーグルトは腸内の健康に良いとされ、医師からも勧められています。その中でも、「ダノンビオ」がプロバイオティクス系のヨーグルトとして人気で、様々なバリエーションが揃っています。女性モデルたちのリアルな声を通じて、お腹の張りに対する悩みや対策が明らかになりました。外見だけでなく、健康にも気を使う女性たちの生活の一端に迫りました。お腹の張りに悩む女性にとって、新たな食事アプローチやプロバイオティクスの活用が一助となるかもしれません。

表 D.15: 人手キーワードにおけるプロンプト 3 の生成例

近年、女性たちの生活スタイルにおいて、恋愛や健康、美容が注目を集めています。これらの要素が結びつき、女性の幸福度向上に寄与するとする新たな提案が浮上しています。NTT ドコモのようなテクノロジー企業が、スマートフォンのアップデートで画面解除の新機能を追加するように、恋愛においても日常生活に新しい彩りを加える方法が模索されています。特に、女性にとっては、恋愛が健康や美容にプラスの影響を与えることが研究で示されています。例えば、アイシェアのアンケートによれば、女性ビジネスパーソンの約 60.6 %が、職場でやる気を出す方法として、軽食やお茶、コーヒーを挙げています。このような気分転換方法が、仕事の効率向上やストレス軽減に繋がるとされ、それが結果的に健康や美容にも良い影響をもたらす可能性があります。さらに、恋愛や結婚においても、幸福度向上に繋がることが研究で明らかにされています。田中将大と里田まいの結婚発表はその一例で、田中は自身のブログで彼女のサポートが成功に貢献したことを述べ、結婚が彼の活躍に期待を寄せる声も上がりました。このような背景から、新たな「恋愛健康法」として、恋愛や結婚を積極的に取り入れ、日常に楽しみや幸福感を取り入れることが提案されています。これは、単なる娯楽や嗜好だけでなく、女性の生活全般にポジティブな影響を与える可能性があり、健康で美しい未来への一環として注目を集めています。今後、企業や研究機関が、恋愛や結婚といった要素を取り入れた新たな商品やサービスの提供に注力することで、女性たちの生活の質を向上させる一翼を担うことが期待されます。女性が豊かな生活を楽しむためには、これらの側面を総合的に捉えたアプローチが求められています。

E GPT-4 を試みた場合の拡張データ

5.4 節において生成した拡張データから「独女通信」の一例を以下の表 E.16 から E.21 に示す.

表 E.16: プロンプト 1 の GPT-4 による生成例 (GPT-3.5 キーワード)

多くの女性が日常的にダイエットに取り組んでいますが、新たなトレンドとして「映画鑑賞ダイエット」が注目されています。この方法は、映画を見ながら楽しくダイエットを行うことを目的としており、特に忙しい女性たちに人気を博しています。最近の研究によると、映画を見ながら行う軽い運動やストレッチは、リラックス効果とダイエット効果を同時に得られることがわかっています。例えば、ロマンチックな映画を見ながら行うヨガやピラティスは、心を穏やかにし、身体のストレスを解放します。また、アクション映画を見ながらの軽いカーディオ運動は、エネルギー消費を高め、効率的なダイエットが期待できます。この新しいダイエット法は、特に女性に人気があります。多忙な日常から離れ、映画を楽しみながら健康的な体を目指すことができるため、ストレスフリーで続けやすいという利点があります。さらに、自宅で気軽にできるため、ジムに行く時間がない人にも適しています。映画鑑賞ダイエットに取り組む女性たちは、SNS を通じてお互いに励まし合ったり、おすすめの映画やエクササイズを共有するなど、コミュニティを形成しています。このような交流は、モチベーションの維持にも寄与しています。専門家は、映画鑑賞ダイエットが身体だけでなく、心の健康にも良い影響を与えることを強調しています。映画のストーリーに没頭することで、日常のストレスから解放され、ポジティブな気持ちになることができるためです。このトレンドは、健康とエンターテインメントの融合として、今後も多くの人々に支持されることが予想されます。映画鑑賞ダイエットは、忙しい現代人にとって、新しいライフスタイルの一部となりつつあります。

表 E.17: プロンプト 2 の GPT-4 による生成例 (GPT-3.5 キーワード)

映画の世界が現実にもたらす影響は計り知れません。最近、女性たちの間で広がっている新しいダイエットトレンドは、意外な場所、すなわち映画館から始まったのです。この現象は、特定の映画に触発された健康的な生活様式と食習慣の変化に焦点を当てています。この流れの発端となったのは、先日公開された映画『フィット・フォー・ライフ』。この作品は、主人公がダイエットと自己受容の旅を通じて変化していく姿を描いています。映画の中で、主人公が実践する健康的な食生活と運動習慣が、観客の心に強く響いたのです。「映画を見た後、自分も健康的な生活を送ろうと思った」と語るのは、25 歳の OL、佐藤さん。彼女のように映画からインスピレーションを受けて、ダイエットや健康的な生活様式を実践する女性が増えています。特に注目されているのは、映画で紹介された「グリーンスムージー」や「ホールフード」など、身体に優しい食品の摂取です。このトレンドは、SNS でも大きな話題となっており、「#フィットフォーライフダイエット」のハッシュタグで多くの女性たちが自分の成果を共有しています。また、この映画の影響を受けた健康食品店やフィットネスクラブも、特別なプログラムを提供し始めているのです。専門家はこの現象を「エンターテイメントが健康意識の向上に貢献している素晴らしい例」と評価しています。映画が単なる娯楽を超えて、人々の生活様式にポジティブな影響を与えることができるのです。しかし、専門家たちは同時に、健康的なダイエットは個々の体質や健康状態に適した方法を選ぶことが重要だと指摘しています。一時的なトレンドに流されることなく、自分に合った健康的な生活様式を見つけることが大切なのです。この映画によって引き起こされたダイエットトレンドは、今後も多くの女性たちに影響を与え、健康への意識を高めるきっかけとなりそうです。

表 E.18: プロンプト 3 の GPT-4 による生成例 (GPT-3.5 キーワード)

#映画で活躍する女優たちのダイエット秘訣
映画業界で輝く女優たちが、彼女たちの健康と美しさを保つために採用しているダイエット方法が注目されています。最近のインタビューで、多くの女優が自然な方法で体型を維持する秘訣を明かしました。特に、ヨーグルトを活用したダイエットが話題となっています。
#ヨーグルトを取り入れたダイエット法
女優たちの間で広がっているのは、ヨーグルトを日常の食生活に取り入れることです。ヨーグルトは、消化を助け、腸内環境を整える効果があり、自然に体重管理ができるとされています。彼女たちの多くは、朝食にヨーグルトを取り入れることで、一日の始まりを健康的に迎えています。
#映画業界における美の基準と健康への意識
映画業界では、見た目の美しさだけでなく、健康も非常に重視されています。女優たちは、厳しいダイエットに頼るのではなく、バランスの取れた食事と適度な運動によって体型を維持しています。ヨーグルトを含む自然食品の摂取は、彼女たちのライフスタイルに欠かせない一部となっています。
#ダイエットの新しい潮流
この傾向は、ダイエット業界にも影響を与えており、多くの女性が映画業界の女優たちに影響を受けています。健康的な体型維持のために、ヨーグルトを日常的に取り入れる女性が増えているのです。
#結論
映画業界の女優たちのダイエット方法は、自然で健康的なアプローチを提唱しています。ヨーグルトを含むバランスの取れた食事と適度な運動は、見た目の美しさだけでなく、内面からの健康をもたらします。これらの方法は、映画業界の外にも広がりを見せており、多くの女性に新たなダイエットの手法として受け入れられています。

表 E.19: プロンプト 1 の GPT-4 による生成例 (人手キーワード)

最近の社会動向において、「女性のキャリアと家庭の両立」というテーマが注目されています。特に、恋愛や結婚において、女性がどのように自分のキャリアを維持し、同時に家庭生活を充実させるかという問題が浮き彫りになっています。この記事では、30 代後半の女性、田中美咲さんの例を挙げ、現代女性が直面している課題と解決策に焦点を当てます。美咲さんは、国際的な広告代理店でキャリアを積み、最近結婚したばかりです。彼女は、「仕事と家庭のバランスは常に難しい」と語りますが、彼女の場合、夫との協力関係が成功の鍵であると強調しています。美咲さんと夫の山田健一さんは、共働きでありながら、家事や育児の責任を平等に分担することで、互いのキャリアと家庭生活の両立を図っています。美咲さんは「夫のサポートがなければ、今の私の成功はあり得なかった」と語り、夫婦間のコミュニケーションの重要性を強調しています。さらに、美咲さんは、恋愛や結婚において女性が自己実現を追求する上で、パートナーの理解とサポートが不可欠であると述べています。彼女は、女性がキャリアを持ち続けることが、個人の幸福だけでなく、家庭の幸福にもつながると信じています。この記事は、現代社会において女性が直面するキャリアと家庭生活のバランス問題に光を当て、共感を呼び起こす内容となっています。美咲さんの例を通じて、多くの女性が自己実現と家庭生活の両立を目指す上で、パートナーの理解と協力がいかに重要かを示しています。

表 E.20: プロンプト 2 の GPT-4 による生成例 (人手キーワード)

最近の独身女性たちは、恋愛と結婚に関する複雑な感情に直面しているようです。今回の調査では、30 代前半の独身女性 100 人を対象に、彼女たちの恋愛観と結婚観について深掘りしました。調査結果によると、約 65 %の女性が「恋愛は楽しいが、結婚には消極的」と回答。多くの女性が恋愛を楽しんでいる一方で、結婚に対しては慎重な姿勢を示しています。結婚に対する慎重な態度の背景には、キャリア、自由、経済的な自立など、さまざまな要因が絡み合っていることがわかりました。また、結婚については「経済的な安定が必要」と考える女性が多く、約 70 %の女性が経済面での不安を理由に挙げていました。特に、自分自身のキャリアを重視する女性たちは、「結婚よりも仕事を優先したい」という意見が目立ちました。一方で、約 30 %の女性は「いずれは結婚したいが、理想の相手に出会っていない」と答えています。恋愛においては理想を追求するものの、現実の相手にそれが見つからないというジレンマに直面しているようです。調査からは、現代女性の恋愛観と結婚観における複雑な心理が浮き彫りになりました。恋愛においては自由でいたいが、結婚には一定の条件を求める。このような矛盾を抱えながらも、それぞれの価値観に基づいて生きる女性たちの姿が見て取れます。今後も、変化する社会の中で女性たちの恋愛観と結婚観は進化し続けることでしょう。彼女たちのリアルな声に耳を傾け、これからの恋愛や結婚の形について考える機会となります。

表 E.21: プロンプト 3 の GPT-4 による生成例 (人手キーワード)

時代が変わる中で、女性たちの恋愛観、結婚観、そして健康と美容への意識も大きく変化しています。現代女性は、より自己実現を重視するようになり、それが恋愛や結婚、そして日常生活における健康と美容のアプローチにも反映されているのです。最近の調査によると、女性たちは恋愛や結婚において、従来の価値観に囚われず、相手との精神的なつながりや共通の価値観を重視する傾向にあります。また、結婚後も自己のキャリアや趣味を大切にする女性が増えており、夫婦間の新しい形が模索されています。健康と美容に関しては、外見だけでなく、内面からの健康美を追求する動きが顕著です。ナチュラル志向のスキンケア製品やオーガニック食品の人気が高まり、健康的なライフスタイルを送る女性が増えています。また、ヨガやピラティスなど、心と体のバランスを整える運動が注目されています。このような変化は、女性たちの自立意識の高まりと深く関連しています。彼女たちは、社会的な期待や固定観念に縛られることなく、自分自身の価値観に基づいて生きることを選んでいます。これは、恋愛や結婚だけでなく、職業選択やライフスタイルにも反映されているのです。これからの時代において、女性たちのこのような自由で多様な生き方は、社会全体に新しい風をもたらすことでしょう。女性一人ひとりが自分らしく輝ける社会を目指して、私たちも応援していきたいと思います。