卒業論文

金型製造の仕上げ工程立案に特化した RAG における LLM の自問自答による精度向上

藤田 将豪

2025 年 2 月 6 日

岐阜大学 工学部 電気電子・情報工学科 情報コース
鈴木研究室

本論文は岐阜大学工学部に
学士（工学）授与の要件として提出した卒業論文である.

藤田 将豪

指導教員：
鈴木 優 准教授

金型製造の仕上げ工程立案に特化した RAG における LLM の自問自答による精度向上*

藤田 将豪

内容梗概

本研究では，LLM(Large Langage Model; 大規模言語モデル) の自問自答により，金型製造の仕上げ工程立案に特化した RAG(Retrieval-Augmented Generation; 検索拡張生成) の精度向上を目的とする．金型製造の仕上げ工程の立案は熟練作業者でないと難しいため，後継者へ技術を提供する手法は今後の工業の持続化に必要不可欠である．既存の単一検索クエリを用いた RAG では，利用者の指示に回答するための内容が文書データ中の複数箇所に存在する場合，参考情報として一度の検索結果のみを使用するため，情報不足が生じてしまうことがある．我々は利用者の指示に回答するための情報不足により，仕上げ工程の全作業を網羅的に回答することが困難であると考えた．そこで本研究では，利用者の指示に回答するための参考情報を補うため，LLM の自問自答に着目した．本研究における LLM の自問自答とは，LLM が検索結果をもとに不足情報を出力し，その不足情報で再び検索を繰り返すことである．我々は LLM の自問自答を繰り返すことにより，1 回目とは異なる検索結果が得られることを期待する．利用者の指示に回答するための情報を補うことができ，仕上げ工程を網羅的に回答できるのではないかと考えた．本研究の評価実験では，三つの金型製造に関する文書データに対して，提案手法と単一検索クエリを用いた RAG の比較を行った．実験結果として，一つの文書データでは提案手法を適用することで，単一検索クエリを用いた RAG と比べ Recall が最大で 0.214 向上したことが確認された．

キーワード

RAG，LLM，自問自答，金型製造，仕上げ工程立案

*岐阜大学 工学部 電気電子・情報工学科 情報コース 卒業論文, 学籍番号:1213033127, 2025 年 2 月 6 日.

目次

図目次

表目次

第 1 章　はじめに

本研究では LLM(Large Language Model; 大規模言語モデル) の自問自答によって，金型製造工場における金型の情報，不具合内容および発生原因などに関する文書データから，金型製造の仕上げ工程の案を自動的に生成する手法を提案する．金型製造の仕上げ工程とは，金型製造の最終段階において製品の精度や仕上がりの品質を高めるために行われる工程である．例えば，部品同士を隙間なく合わせる作業や複数の部品を組んで同時加工をするための組付作業である．

金型製造工場では仕上げ工程の立案を熟練作業者が担っているため，近年の作業者の高齢化により，熟練作業者から若手作業者への技術の継承が大きな課題となっている．また，人手で仕上げ工程に関する文書の中から必要な情報を抽出し適切な仕上げ作業を決めるには手間がかかってしまい，作業者にとって負担になる．このような状況下で，仕上げ工程の立案を支援するための手法の開発は必要不可欠であると考えられる．このような経験のない作業者でも仕上げ工程を立案するために，これまで蓄積された社内情報を知識ベースとして LLM を活用した精度の高い RAG(Retrieval-Augmented Generation; 検索拡張生成)[1] を構築する．RAG とは，LLM と情報検索技術を組み合わせた手法である．RAG ではテキスト検索システムから利用者の指示に回答するための参考情報を検索し，それらの情報をもとに LLM を用いて回答を生成する．本研究でのテキスト検索システムとは，社内で管理されている金型製造に関する文書データから検索を行う．RAG により LLM が事前学習データに含まれていない情報にも対応できるため，社内情報から抽出した参考情報を活用する手法として有効である [2]．これにより，若手作業者の意思決定を補助するための適切な仕上げ工程の案を提示するシステムの実現を目指す．

LLM は教師データとなるテキストデータをもとに，質問応答や文章生成などのタスクを実行できる推論能力を備えた技術である．我々は，LLM の事前学習データには本研究で扱う金型製造に関する社内情報が含まれていないため，仕上げ工程の案を適切に生成することが難しいと考えた．

既存の RAG では，単一の検索クエリにより一回の検索結果のみを利用者の指示に回答するための参考情報とする．検索クエリとは，テキスト検索システムの入力を指す．具体的には，仕上げ工程の立案に関する「不具合内容，発生原因，対策内

容」を検索クエリとして用いる．そこで我々は，検索結果が参考情報として利用者にとって不十分な場合でもそれを補完する仕組みがなく，重要な情報が欠落する問題点があると考えた．我々は，この問題点により必要な情報を網羅的に取得できず，利用者の指示に対して網羅的な回答を生成することが難しいと考えた．本研究における網羅的な回答とは，仕上げ工程に必要な作業の案を漏れなく提案することである．

本研究では，LLM の自問自答を活用することによって既存の RAG の課題を解決し，参考情報の情報不足を補う手法を提案する．我々は LLM が検索結果の不足情報を特定し，その不足情報を補うための検索クエリを生成することによって，利用者の明示的な指示がなくても情報不足を解決できるのではないかと考えた．例えば，検索結果において「No.8 不具合内容：」というように取得されたとする．そこで，我々は LLM がこの検索結果に対して「No.8，対策内容」というような検索クエリを生成し，検索結果にはなかった「No.8 の対策内容」に関する新しい検索結果が取得できることを期待する．また，LLM が生成した検索クエリを用いて再検索を行うことで，単一検索では得られなかった情報を補完し，検索結果の網羅性が向上するのではないかと考えた．本研究における検索結果の網羅性とは，検索結果に網羅的な回答をするための重要な参考情報が含まれているか否かである．

提案手法では，まず利用者の入力した検索クエリに対して検索を行う．次に，検索結果をプロンプトとして LLM に入力し，不足情報に関する検索クエリを生成する．次に，不足情報に関する検索クエリを用いて検索を行う．最終的に，全ての検索結果を統合して LLM が利用者の指示に対する回答を生成する．本研究における LLM の自問自答とは，LLM が自ら不足情報の検索クエリを生成し，回答を得るまでのプロセスを指す．我々は，LLM の自問自答を繰り返すことで，RAG における参考情報の情報不足が解消され，LLM の回答精度を高めることを期待する．

我々は本手法の有効性を検証するために，ベースラインとして単一の検索クエリを用いた RAG との比較実験を行った．実験では，三つの文書データを使用し，それぞれの内容に従って三つの仕上げ工程を立案した．実験の結果，一つの文書データにおいて単一の検索クエリを用いた RAG と比べ Recall が最大で 0.214 向上したことが確認された．Recall が向上したことで，ベースラインよりも網羅的な回答を生成できるようになったと言える．また，対応のある 2 標本 t 検定を行った結果，

一つの文書データにおいては提案手法における LLM の自問自答を 2 回行った場合に有意差が確認された．

本稿による貢献は以下の通りである．

- 提案手法を適用した場合，既存手法に比べて Recall が向上し，仕上げ工程の立案における有用性を示した．
- RAG における LLM をファインチューニングしなくても本手法における LLM の自問自答を適用することで，Precision，Recall，F 値の三つの指標が向上することを示した．

本稿の構成は以下の通りである．2 章では，本研究で用いた基本的な技術や手法について述べる．3 章では，本研究の関連研究について述べる．4 章では，LLM の自問自答と RAG を組み合わせた提案手法について述べる．5 章では，評価実験について述べる．6 章では，本研究のまとめと今後の展望について述べる．

第 2 章　基本的事項

2.1　LLM

LLM(Large Language Model; 大規模言語モデル) とは，ニューラルネットワークにおける大規模なパラメータを用いて事前学習したモデルである．事前学習では与えられた単語から確率分布に基づいて，次に並ぶ単語として最も確率の高い単語を予測する．例えば「空は」と入力された時,「青」の予測確率が 0.75,「赤」の予測確率が 0.1 とすると,「空は青」と出力される可能性が高いと言える．LLM は主に質問応答や機械翻訳のような自然言語処理や画像処理，音声処理の分野で幅広く使われている．代表的なモデルとして，Open AI 社の GPT(Generative Pretrained Transformer)，Meta 社の Llama(Large Language Model Meta AI) などが挙げられる．これらの LLM の仕組みには Transformer と呼ばれるモデルが基盤となっている．Transformer については 2.2 節で説明する．

2.2　Transformer

Transformer[3] とは，2017 年に Vaswani らによって提案された系列変換モデルである．系列変換とは，機械翻訳のようにある文字列を別の文字列に変換することである．これまで RNN が使われてきたが，長文の処理性能が低いという問題点があった．RNN とは再帰型ニューラルネットワークで，主に文字列のような系列データを扱うために利用されている．RNN では，入力文を逐次的に処理し一つの固定長のベクトルで表現される．そのため，長文になると処理の過程で重要な情報が失われる可能性があり，入力文全体の意味を捉えることができないという問題がある．RNN の問題点を解決するため，Transformer は Attention 層のみを用いたニューラルネットワークとなっている．

Transformer は，エンコーダとデコーダの二つの部分から構成されている．エンコーダは入力系列の特徴を抽出し，デコーダは系列を生成するという役割を担っている．エンコーダは 6 層で構成されており，6 層とも同じ構造である．各層では，Multi-Head Attention 層と全結合層の二つで構成されている．デコーダでも同じ

く 6 層で構成されており，6 層とも同じ構造である．ただし，各層ではエンコーダの二つの層の間にエンコーダの出力を受け取る Multi-Head Attention 層を追加した形になっている．

2.3 Attention

Attention とは，文中のある単語が他の単語との関係性をどの程度重要視すべきかを表すスコアのことである．Transformer において Attention を採用することには利点がある．例えば, 複数の単語間の関係を同時に計算できるため, 逐次的に処理する RNN よりも計算効率が高い．Attention のスコアは主に Scaled Dot-Product Attention と呼ばれる手法で計算される．

2.3.1 Scaled Dot-Product Attention

Scaled Dot-Product Attention は，Query と Key と Value の三つのベクトルで計算される．以後，Query と Key と Value をそれぞれ $\mathbf{Q}$，$\mathbf{K}$，$\mathbf{V}$ と表す．

$\mathbf{Q}$，$\mathbf{K}$，$\mathbf{V}$ はエンコーダの最初の段であれば，入力した単語のベクトル $\mathbf{X}$ にそれぞれ重み $\mathbf{W}^Q$，$\mathbf{W}^K$，$\mathbf{W}^V$ を掛け合わせて，$\mathbf{X}\mathbf{W}^Q$，$\mathbf{X}\mathbf{W}^Q$，$\mathbf{X}\mathbf{W}^V$ と表される．エンコーダの二段目以降ならば，前段の出力にその段の特有の別の重みを掛けることで計算される．

Scaled Dot-Product Attention では，$\mathbf{Q}$ と $\mathbf{K}$ の内積により二つのベクトルの関連度を計算し，$\sqrt{d_k}$ で割った後に，softmax 関数を適用する．この時，$\mathbf{Q}$ と $\mathbf{K}$ は，次元 $\sqrt{d_k}$ を持ち，$\mathbf{V}$ は次元 $\sqrt{d_v}$ を持つ．

Scaled Dot-Product Attention は，以下の式 2.3.1 で計算される．

$$\mathrm{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V} \tag{2.3.1}$$

2.3.2 Multi-Head Attention

Multi-Head Attention とは，Scaled Dot-Product Attention を一つの head とみなした時，複数 head を並列化したものである．各 head が独立して学習することで，異なる視点や依存関係を捉えることができモデルの表現力が向上する．また，単一の Attention よりも Multi-Head Attention を用いた方がモデルの精度や文脈の理解能力が向上することが確認されている．

Multi-Head Attention は，以下の式 2.3.2 で計算される．

$$\mathrm{MultiHead}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathrm{Concat}(\mathrm{head}_1,\ldots,\mathrm{head}_h)\mathbf{W}^O \tag{2.3.2}$$

$$where \quad head_i = \mathrm{Attention}(\mathbf{Q}\mathbf{W}_i^Q,\mathbf{K}\mathbf{W}_i^K,\mathbf{V}\mathbf{W}_i^V) \tag{2.3.3}$$

2.4 RAG

RAG(Retrieval-Augmented Generation; 検索拡張生成)[1] とは 2021 年に Lewis らによって提案された，LLM による回答生成と情報検索を組み合わせた手法である．RAG の主な目的は，LLM が外部の情報を参照することによって，専門的な知識などの事前学習データに含まれていない情報について回答することである．本研究では，金型製造に関する文書データを用いて RAG を構築する．

RAG は大きく Retriever と Generator で構成されており，これらについて以下で説明する．

2.4.1 Retriever

Retriever は，RAG における情報検索の役割を担っている部分である．具体的には，利用者の入力に基づいて，類似度が高いチャンクの内容を文書データから検索し出力する．チャンクとは，大きな文書を扱いやすい単位に分割した文章の塊で，1 つ以上のトークンから構成される．また，トークンとは自然言語処理においてテキストを分割した最小単位のことである．Retriever で検索する際には，利用者の入力との類似度を計算するために事前に文書データをベクトル化しておく必要があ

る．文書データをベクトル化する際に，文章全体をチャンクに分割する．チャンクに分割する理由として，Generator での入力トークン数に制限があるためである．従って，文書データの内容が入力トークン数の制限を超えている場合，利用者の質問や指示に回答するための参考情報として Generator に入力できないことになる．Retriever の処理手順を以下に示す．

1. 利用者の入力文をベクトル化する．
2. 事前にベクトル化された文書データのチャンクと入力文との類似度を計算する．
3. 類似度が高いチャンクの内容を出力する．

本研究では検索速度の向上のために ANN(Approximate Nearest Neighbor; 近似最近傍探索)[4] を用いた．ANN については 2.6 節で説明する．また，類似度を計算する際の距離関数はコサイン類似度を用いた．コサイン類似度については 2.7 節で説明する．

2.4.2 Generator

Generator は，Retriever で取得されたチャンクの内容を参考情報として，利用者の質問や指示に対して回答生成を担っている部分である．Generator には LLM を用いるのが一般的である．Generator の処理手順を以下に示す．

1. Retriever から出力されたチャンクを参考情報として，利用者の質問や指示と組み合わせてプロンプトを構築する．
2. 構築されたプロンプトを LLM に入力し，モデルによる推論結果を回答として出力する．

2.5 埋め込み

埋め込みとは，文や単語を固有の数値ベクトルで表すことである．埋め込みベクトルは次元空間で表現され，文や単語の類似度を計算するために利用される．主

な埋め込みモデルには，text-embedding-3-small や intfloat/multilingual-e5-large が挙げられる．これらの埋め込みモデルは，Transformer をベースとしてテキストをベクトル化している．例えば intfloat/multilingual-e5-large では，Transformer のエンコーダ部分を利用している．埋め込みモデルの入力文は単語や句読点，または一文字のトークンに分割される．エンコーダ部分の出力は入力文の各トークンの埋め込みベクトルが得られるが，文全体の埋め込みベクトルではない．文全体の埋め込みベクトルを得るのに平均プーリングを使う．平均プーリングとは，全てのトークンに対応するベクトルを足し合わせ，トークン数で割ることで文の埋め込みベクトルとして利用する．本研究では，金型製造に関するテキストデータと利用者からの入力文に対し，埋め込みモデルを用いて文章を埋め込み表現に変換する．

2.6 ANN

ANN(Approximate Nearest Neighbor; 近似最近傍探索)[4] とは，高次元空間におけるデータ間の類似性を高速に探索するための手法である．ANN はデータセット内の特定のクエリに対して，一番近いデータを正確に見つけるのではなく，近似的に近いデータを高速に見つけることを目的としている．本研究では，ANN を用いることで RAG における Retriever の検索速度を高速化し，大規模な文書データでの計算コストの削減を目指す．

2.7 コサイン類似度

コサイン類似度とは，二つのベクトル間の類似性を測るための指標の一つであり，特に高次元ベクトル空間で有効である．コサイン類似度の値は，$[-1, 1]$ の範囲をとり，値が 1 に近いほど高い類似性を示す．コサイン類似度は，ベクトルの長さの影響を受けない点が特徴である．また，テキストデータにおいては意味的な類似性を評価できることから情報検索において使われている．

二つのベクトル $\mathbf{a}$ と $\mathbf{b}$ のコサイン類似度を式 2.7.1 に示す．

$$\cos(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|} \tag{2.7.1}$$

2.8 評価指標

本研究では，システムの性能を評価するために Precision，Recall，F 値の三つの指標を使用する．これらを説明するにあたり，表 2.1 の混同行列を用いる．本稿において, 正解の仕上げ工程における作業を正解作業, モデルによって提案された仕上げ工程における作業を提案作業と呼ぶ．従って，表中の正解データの Positive は正解作業で，予測データの Positive は提案作業である．また，正解データの Negative とは正解作業ではない作業で，予測データの Negative とは提案作業でない作業を示す．

True Positive とは，提案作業が正解作業と一致しているものである．False Positive とは，正解作業でない作業を提案作業としたものである．False Negative とは，提案作業に含まれていない正解作業である．True Negative とは，正解作業にも提案作業にも含まれていないものである．

2.8.1 Precision

Precision とはモデルが Positive と予測したデータのうち，実際に Positive であるデータの割合を示す．本研究では，提案作業のうち正解作業である割合を指す．

Precision は以下の式 2.8.1 で計算される．

$$Precision = \frac{TP}{TP + FP} \tag{2.8.1}$$

表 2.1　混同行列

		正解データ	
		Positive	Negative
予測データ	Positive	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)

2.8.2 Recall

Recall とは実際の Positive であるもののうち，モデルが Positive と正しく予測できた割合を示す．本研究では，正解作業のうち正しい提案作業の割合を指す．

Recall は以下の式 2.8.2 で計算される．

$$Precision = \frac{TP}{TP + FN} \tag{2.8.2}$$

2.8.3 F 値

F 値とは，Precision と Recall の調和平均を計算し，両方のバランスを評価する指標である．F 値を高めるためには，Precision と Recall をバランス良く高める必要がある．本研究では，システムの全体的な性能を評価するために F 値を使用した．

F 値は以下の式 2.8.3 で計算される．

$$F = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{2.8.3}$$

2.9 対応のある 2 標本 t 検定

対応のある 2 標本 t 検定とは，異なる二つの条件で同じ対象から得られたデータを用いて，二つの条件間の平均値に有意な差があるかを検証する統計手法である．本研究では，同じ文書データを用いてベースラインの手法と提案手法の結果を比較するため，対応のある 2 標本 t 検定を使用した．

本研究における対応のある t 検定は以下の二つの仮説を検証する．

1. 帰無仮説 H_0：二つの手法間における F 値の平均値に有意な差がない．
2. 対立仮説 H_1：二つの手法間における F 値の平均値に有意な差がある．

次に，有意水準 α を設定する．有意水準とは，帰無仮説が正しいくないと判断する基準である．本研究では有意水準 $\alpha = 0.05$ とした．

次に，検定統計量 T を求める．検定統計量 T の算出方法を式 2.9.1 に示す．ここで，$\bar{d}$ は得られた結果の二群間の差の平均，${s_d}^2$ は不偏分散，n は自由度である．

$$T = \frac{\bar{d}}{\sqrt{{s_d}^2/n}} \tag{2.9.1}$$

次に，検定統計量 T を用いて p 値を求める．p 値は，検定統計量 T が t 分布のどこに位置するかを評価し，その位置に対応する確率を値とする．ここで，得られた p 値と有意水準 α を比較する．p 値が有意水準 α より小さい場合，帰無仮説が棄却され，統計的に有意な差があると言える．一方で，p 値が有意水準 α より大きい場合，帰無仮説が採択される．

第 3 章　関連研究

自問自答を用いた LLM の性能を向上させるための研究は，現在までに多く行われている．Wei ら [5] は Chain-of-Thought と呼ばれる手法を提案した．この手法では，問題の最終回答に至る一連の中間推論ステップが含まれたプロンプトを言語モデルに与えることで，モデルが解答を導く過程を可視化しながら正解に至ることを目的としている．Chain-of-Thought は，数学のような論理的なタスクで精度が向上することを示した．

Press ら [6] は言語モデルの構成的推論能力を向上させるために self-ask という手法を提案した．構成的推論能力とは，モデルが複数のサブ質問を解く必要があるタスクを正確に推論できる能力である．self-ask では，二つの質問で構成されている一つの質問に対してより単純なサブ質問に分解し，次にサブ質問に答え，最後にメイン質問に答える．self-ask を用いた実験では，従来の Chain-of-Thought よりも Accuracy が高いことが確認された．

Wang ら [7] は Chain-of-Thought の問題点を解決するために，Strategic Chain-of-Thought という手法を提案した．従来の Chain-of-Thought では，LLM が中間推論の段階で誤った判断を出力してしまうと，その後のステップにも影響が出てしまい，最終的に不正確な回答が導かれることがある．また，推論ステップが増えるほど間違った判断による誤差が蓄積されやすく，Chain-of-Thought の一貫性が問題となることもある．Strategic Chain-of-Thought では，LLM が問題を解決するための最適な戦略を特定してから段階的に推論を行い，最終的な回答を出力する．

また，RAG の精度を向上させるための研究も多く行われている．Asai ら [8] は従来の RAG よりも回答性能を高めるために，Self-RAG という手法を提案した．Asai らは，質問の回答に RAG が必要かどうかの判断と外部知識を元に LLM が生成した回答が正しいかどうかの判断に着目した．この手法では，誤っていると判断された内容は回答に利用されないような仕組みである．この手法を用いた実験により，長文生成や選択式の推論タスクにおいて従来の RAG よりも精度の高い結果が確認された．

Yan ら [9] は RAG よって取得された文書が不正確だった場合の性能低下を改善するために，CRAG という手法を提案した．CRAG では検索結果を「正確」,「不

正確」,「曖昧」に分類し,「不正確」と「曖昧」の時に Web 検索を通じて追加情報を取得する．この手法により，そもそも取得された文書が間違っていた場合に知識を補うことができる．この手法を用いた実験により，長文生成や選択肢の推論タスクにおいて Self-RAG よりも精度が上がったことが確認された.

Jeong ら [10] は Adaptive-RAG という手法を提案した．Adaptive-RAG では利用者からの質問について,「簡単」,「中程度」,「複雑」の三つに分類し，それぞれの場合で最適な手法を適用させた．この手法では，簡単な質問は LLM のみで回答し，中程度の質問は LLM と一回の検索を組み合わせて回答し，複雑な質問は LLM と複数回の検索を繰り返し，推論を重ねて回答をする．この手法を用いた実験により，Self-RAG と比較し，全てのデータセットにおいて精度が上がったことが確認された.

先行研究 [5][6][7] より，LLM の自問自答を活用することでモデルの精度が高くなることが示された．従って，本研究では LLM の自問自答を RAG における検索段階で活用することで，RAG の性能も良くなるのではないかと考えた．また，本研究と RAG に関する先行研究 [8][9][10] との相違点は，検索結果に対して LLM の自問自答を用いることで検索結果の情報不足を補完することを目的としている点，金型製造のような特定のドメインのタスクを対象としている点である.

第 4 章　提案手法

本手法では，LLM の自問自答を活用することで網羅的な回答生成が可能な RAG の手法を提案する．LLM の自問自答とは，LLM 自身が質問を生成し，その質問に回答することである．本手法では LLM が自ら不足情報ついての検索クエリを生成し，利用者の指示に対して最終的な回答を得るまでのプロセスを指す．本手法は以下の六つのステップから構成される．

Step 1 利用者の入力した検索クエリに基づき，初回の検索を行う．検索クエリは仕上げ工程の立案に関連する「不具合内容，発生原因，対策内容」を用いる．

Step 2 前の段階における検索結果をプロンプトとして LLM に入力し，検索結果に基づいて不足している情報に関する検索クエリを生成する．例えば，利用者の目的を「仕上げ工程の立案」とし，指示文を「以下の参考情報における不足情報について出力して下さい」とする．最後に，参考情報として前の段階における検索結果を記載する．

Step 3 Step 2 で生成された検索クエリを用いて再度検索を行う．Step 2 で「発生日」が出力された場合,「発生日」を検索クエリとして Retriever に入力する．

Step 4 Step 2 と Step 3 を繰り返し，得られた全ての検索結果を統合して参考情報とする．例えば,「No.1」に関する内容と「No.3」に関する内容が得られたとすると，それらを合わせて一つの参考情報とする．

Step 5 利用者の指示と Step 4 で得られた参考情報でプロンプトを構築する．例えば，例えば,「仕上げ工程を立案して下さい」という指示文の後に，Step 4 で作成した参考情報を記載する．

Step 6 Step 5 のプロンプトを用いて LLM に入力し，最終的な回答を得る．例えば,「トライアウト，部品合わせ，まとめ」という仕上げ工程が箇条書きで出力される．

これらの手順の詳細について各節で説明する．

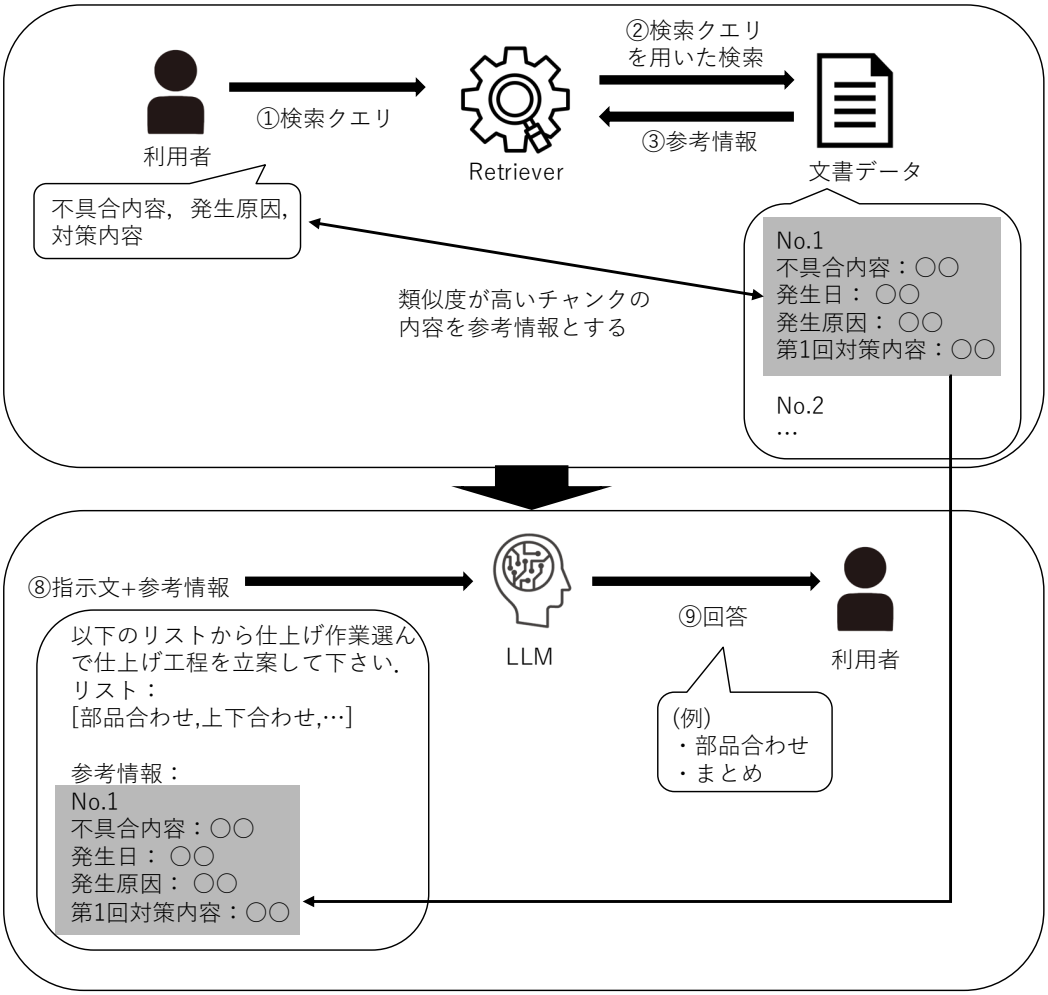

図 4.1　単一の検索クエリを用いた RAG の概要図

4.1　RAG の構築

本節では，本研究における単一の検索クエリを用いた RAG の構築方法について説明する．以下に提案手法との比較として単一の検索クエリを用いた RAG の処理手順を示す．

Step 1 利用者の入力した検索クエリに基づき，初回の検索を行う．検索クエリは仕上げ工程の立案に関連する「不具合内容，発生原因，対策内容」を用いる．

Step 2 利用者の指示と Step 1 で得られた検索結果を参考情報としてプロンプトを構築する．例えば，「仕上げ工程を立案して下さい」という指示文の後に，検索結果を記載する．

Step 3 Step 3 のプロンプトを用いて LLM に入力し，最終的な回答を得る．例えば，「部品合わせ，まとめ」という仕上げ工程が箇条書きで出力される．

本手法における単一の検索クエリを用いた RAG の概要図を図 4.1 に示す．本研究では，一つの仕上げ工程を立案するのにそれに対応した一つの文書データを用いる．図 4.1 では，利用者から入力された「不具合内容，発生原因，対策内容」という検索クエリを用いて検索を行う．例として，「No.1」が含まれているチャンクとの類似度が高かったとすると，このチャンクの内容を利用者の指示に回答するための参考情報とする．次に，「仕上げ工程を立案して下さい」という利用者の指示と参考情報を組み合わせて LLM に入力し，回答を生成する．

4.1.1 ベクトルデータベースの構築

RAG では，Retiever における類似度検索をするために，利用者の入力した検索クエリと文書のテキストデータをベクトル化する必要がある．そこで我々は，テキストの埋め込み表現を得るため，Hugging face で公開されている埋め込みモデルを用いた．本手法では 300 トークンを一つのチャンクとして文書内のテキストを分割した．また，チャンク間のオーバーラップを 50 トークンに設定した．オーバーラップとはテキストの文脈を保持するため，各チャンクにおいて前後の一部を重複するようにすることである．その後，埋め込みモデルを用いて各チャンクごとのベクトルを生成した．

本手法では FAISS(Facebook AI Similarity Search) というライブラリを使用した．FAISS とは，大規模なベクトルデータセット上での近似最近傍検索を効率的に実行するためのライブラリである．ベクトル化されたデータを FAISS に登録し，近似最近傍検索を用いた高速検索可能なインデックスを構築した．埋め込みモデルによって生成された高次元のベクトルをインデックス化することで，大量のデータから効率的に検索できることが期待される．

4.1.2 Retriever の設定

本手法では，参考情報を抽出する際にコサイン類似度を用いた検索を行う．本研究では金型製造の仕上げ工程を立案するという目的があるため，仕上げ工程に関する検索クエリを選択する必要がある．従って，Retriever の入力を「不具合内容．発生原因，対策内容」とした．また，検索により参考情報として取得するチャンクの数を 3 に設定した．Retriever に入力された検索クエリのベクトルを $\vec{x}$，分割されたチャンクのベクトルを $\vec{y}$ としてコサイン類似度の計算式を式 4.1.1 に示す．

$$\cos(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{\|\vec{x}\|\|\vec{y}\|} \tag{4.1.1}$$

4.1.3 Generator の設定

Generator には LLM を用いる．最終的な回答として仕上げ工程の案を出力するため，LLM に入力する具体的なプロンプトを図 4.2 に示す．ここで，{context}には RAG における Retriever による検索結果が入力される．このプロンプトは，作業者が LLM の立案を見やすいように仕上げ工程を箇条書きにすること，不要な情報は出力しないこと，立案した作業を選んだ理由を出力することを考慮して構築した．

4.2 LLM の自問自答

本節では，本手法における LLM の自問自答について説明する．本手法の概要図を図 4.3 と図 4.4 に示す．図 4.3 では，不足情報に関する検索クエリを用いて新しい参考情報を得るまでのプロセスを表している．図 4.4 では，最終的な回答を得るまでのプロセスを表している．例として，単一の検索クエリを用いた RAG における検索で得られた参考情報の中に「No.1」に関する内容しかないため，不足情報に関する検索クエリが「No.2，不具合」と出力されたとする．出力された検索クエリを用いて検索した結果，「No.2」が含まれているチャンクの内容との類似度が高く新しい参考情報として取得する．最後に，単一の検索クエリを用いた RAG におけ

参考情報をもとに，ユーザからの指示にできるだけ正確に答えてください.
出力形式を必ず守ってください:
1. 選ばれた仕上げ作業の順番のリスト（箇条書き）.
2. 不要な情報は出力しない
3. できればそのように選んだ理由も出力.

参考情報：
{context}

ユーザからの指示：
必ず以下のリストの中から適切な仕上げ作業名を選んで仕上げ工程を立案して下さい.
リスト：
[部品合わせ, 上下合わせ, 入駒合わせ, 傾斜合わせ, 直上合わせ,
スライド合わせ, 部品合わせ見直し, 手加工, 上面ナラシ, ベント手加工,
磨き 仕上課, 共 SET, シャフト調整, 型割, 反転, 可動バラシ, 可動組付,
固定バラシ, 固定組付, 冷却, 洗浄, まとめ, 窒化準備, 窒化組付, シボ準備,
HR,HR 以外電気, 分解点検 (自主), 仕様変更, 仕様確認, ペイント, 外観,
写真, 溶接, 出張, 成形機内作業, ブラスト, チェックシート作成,
分解点検 (立会い), 足回り準備, メンテナンス, 梱包,MC 共加工,
MC EP 加工,EDM 加工,EDM 共加工,EDM ピカ加工, 大連部品着,
足まわり着, 金型一式入荷, 可動型一式入荷, 測定, シボ出し, シボ完,
窒化出し, 窒化完]

図 4.2　仕上げ工程を立案するためのプロンプト

る検索で得られた参考情報と不足情報に関する検索クエリを用いて取得した新しい参考情報を組み合わせて，利用者の指示に対し回答をする．本研究における不足情報を用いた検索を繰り返す場合，新しい参考情報を前の段階における参考情報に加

えてから，同じように不足情報に関する検索クエリを生成する．

本手法では，次の二つの効果を期待する．一つ目は，LLM の自然言語処理能力を活用することで，利用者の目的と検索結果に基づいて不足している情報を特定し，それに関する検索クエリを生成できることである．このプロセスにより，利用者が明示的に指定しなくても不足情報を補完することが可能となる．二つ目は，LLM の自問自答によって生成された検索クエリを用いて再度検索を行うことで，単一クエリでは得られなかった異なる文脈や観点を含む検索結果を取得できる．これにより，回答生成に必要な情報の網羅性が向上し，網羅的な回答を得られる可能性が高まる．このように，LLM の自問自答は，RAG における検索結果の情報不足を補い，利用者の指示に対してより網羅的な回答を生成する上で有効であると考えられる．

我々は本手法において，LLM が検索クエリを生成する上で出力形式に守る必要があると考えた．我々が設計したプロンプトを図 4.5 に示す．このプロンプトは，利用者の目的と与えられた参考情報{retrieved_info}から目的に対する不足情報を特定し，その不足情報について日本語の検索クエリを生成するように設計されている．また，我々は不足情報についての検索クエリのみを出力することによって，出力結果をそのまま Retriever に入力できるように設計した．

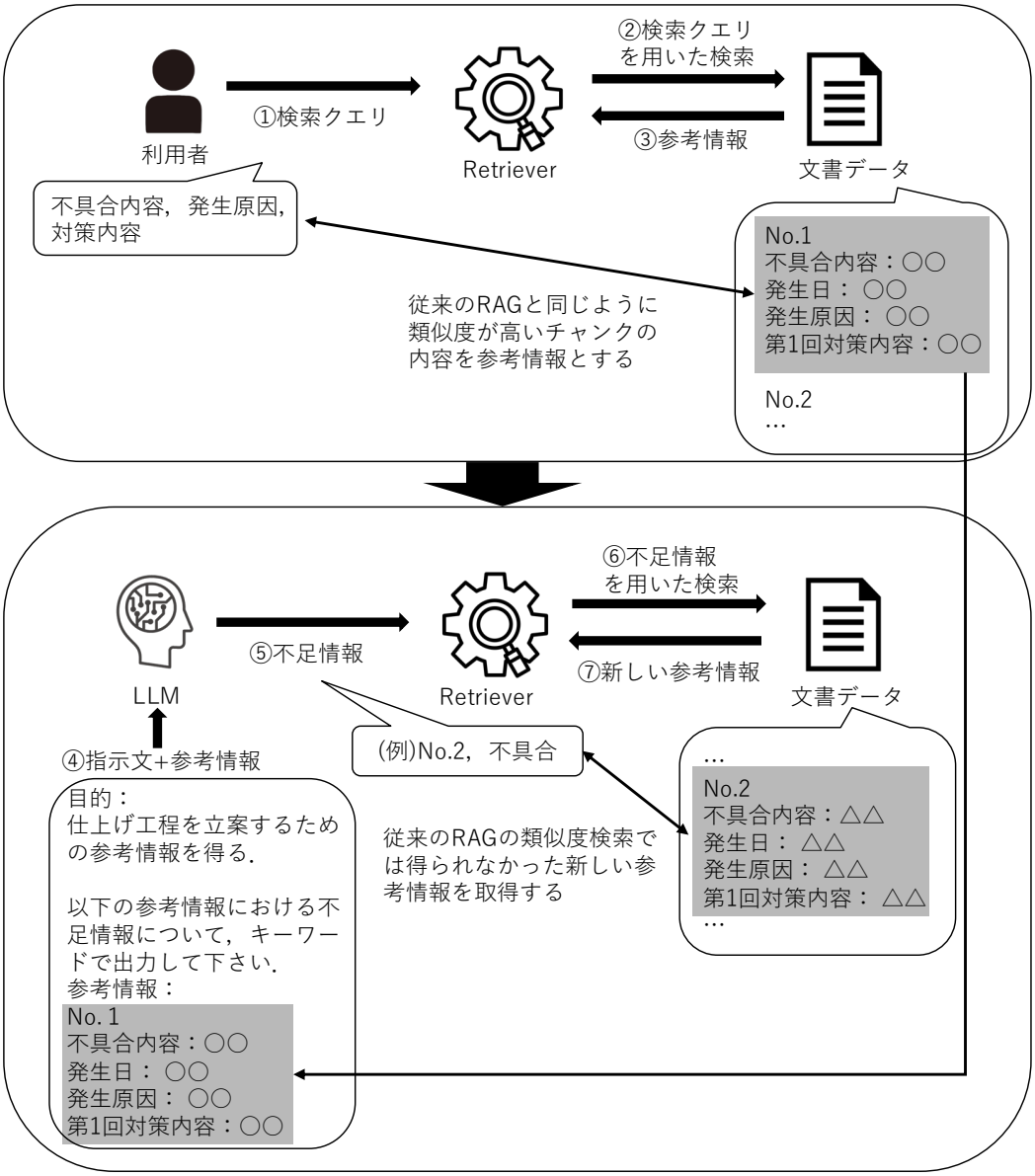

図 4.3　提案手法の概要図 1

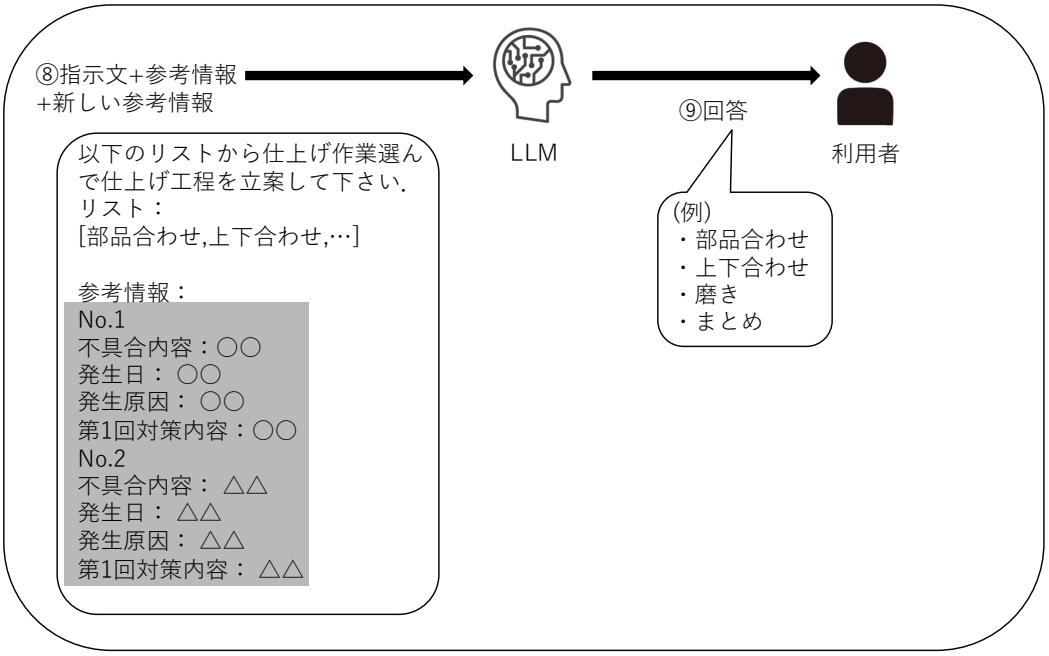

図 4.4　提案手法の概要図 2

以下の目的と参考情報をもとに，ユーザからの指示にできるだけ正確に答えて下さい.

目的：
仕上げ工程を立案するための参考情報を取得する.
参考情報：
{retrieved_info}

指示：
この参考情報について不足情報を特定し，それについて日本語のキーワードを考えて下さい．ただし，複数個でも良い.
不要な情報は出力せず，キーワードだけ出力して下さい.

図 4.5　不足情報についての検索クエリを生成するプロンプト

第 5 章　評価実験

本実験では，金型製造に関する文書データを用いて自動的に仕上げ工程の立案を行い，提案手法による RAG システムの精度を評価する．本実験の目的は二つある．一つ目は，不足情報を用いた検索を 1 回行った場合，単一の検索クエリを用いた RAG よりも精度が高いかどうかを検証することである．二つ目は，不足情報を用いた検索回数によって精度が向上するかどうかを検証することである．

5.1　使用データ

本実験では，株式会社黒田製作所から提供された三つの文書データを使用する．これらの文書データには金型製造に関する不具合内容，発生原因，修正の指示内容などが記載されている．また，正解作業として同社から提供された「指示書管理 No」ごとの仕上げ工程のデータを使用する．正解作業とは，実際の仕上げ工程における作業を指す．このデータには，正解作業に加えて仕上げ作業名の候補として 63 種類の作業が含まれており，本実験ではこれらの候補から仕上げ作業を選択し，仕上げ工程の立案を行う．仕上げ作業名の候補は図 4.2 のリストに記載されている．

5.2　実験手順

本実験では，提案手法とベースラインとして単一の検索クエリを用いた RAG との精度を比較することによって，本手法の有効性を評価する．ベースラインとして使用する RAG は，4.1 節で述べた手順に従って構築した．埋め込みモデルには，intfloat/multilingual-e5-large*を使用し，LLM には elyza/Llama-3-ELYZA-JP-8B†を使用した．本実験では評価指標として，Precision，Recall，F 値を用いる．また，提案手法を用いた RAG の F 値の変化に有意な差があるかを調べるために，対応のある 2 標本 t 検定を行う．実験手順については以下の通りである．

*https://huggingface.co/intfloat/multilingual-e5-large
†https://huggingface.co/elyza/Llama-3-ELYZA-JP-8B/tree/main

Step 1 提供された PDF の文書データをテキストに起こし，テキストデータとして扱う．このテキストデータから仕上げ工程を立案するための参考情報を取得する．

Step 2 4.1 節で述べた手順に従ってベースラインの RAG を構築し，提案作業と正解作業を比較し，Precision，Recall，F 値を算出する．

Step 3 提案手法における不足情報を用いた検索を k 回適用した RAG を構築し，提案作業と正解作業を比較し，Precidion，Recall，F 値を算出する．

Step 4 Step 2 と Step 3 で算出した F 値を用いて対応のある 2 標本 t 検定を行う．

上記の手順で本実験における RAG の精度を評価する．ただし，Step 4 では，10 回分の出力から算出した F 値の平均値を用いる．また，提案作業とはモデルによって提案された仕上げ工程における作業を指す．

5.3 結果・考察

本節では各文書データを用いた実験の結果を示す．我々は仕上げ工程を立案する際に，Precision よりも Recall が高い方が望ましいと考えた．Recall が高い方が望ましい理由として二つ挙げられる．一つ目は，仕上げ工程で必要な作業が提案されなかった場合，製品に問題が発生する可能性があるためである．工程の網羅性が求められるため，Recall が高ければ必要な作業が提案された作業に多く含まれることになり，このようなリスクを低減することができる．二つ目は，Precision が低い場合の影響が小さいためである．Precision が低く Recall が高い場合，不要な作業が提案に含まれることがあるが，これらは後から人間の確認によって取り除くことが可能である．以上より，本実験では Recall が向上した方が仕上げ工程の立案に望ましい結果だと言える．

5.3.1 指示書管理 No.23102415

「指示書管理 No.23102415」における仕上げ工程の立案を行った際の実験結果及び検定結果を表 5.1 に示す．表 5.1 では，提案手法を適用したことでベースラインと比べ Precision が最大 0.034，Recall が最大 0.214，F 値が最大 0.128 向上した．

表 5.1　5.3.1 節におけるベースラインと提案手法の評価指標比較と p 値

手法	**Precision**	**Recall**	**F 値**	p **値**
ベースライン	0.258	0.272	0.225	-
提案手法 ($k = 1$)	0.287	0.414	0.295	0.066
提案手法 ($k = 2$)	**0.292**	**0.486**	**0.353**	**0.014**
提案手法 ($k = 3$)	0.204	0.472	0.237	0.818

Recall がベースラインよりも向上したことより，仕上げ工程の立案には望ましい結果となった．提案手法 ($k = 2$) において，Precision，Recall，F 値の三つの指標とも最も高かった．また，提案手法 ($k = 3$) において，Precision のみベースラインのの値を下回ったことが確認された．対応のある 2 標本 t 検定を行った結果，提案手法 ($k = 2$) のみ p 値が有意水準 0.05 を下回り，F 値の平均値に有意な差が確認された．

我々は提案手法 ($k = 3$) において Precision が下がった原因として，複数回の検索で増加した参考情報に含まれる重複した情報が影響を与えていると考えた．本実験では参考情報の重複について考慮していなかったため，仕上げ工程の立案には不要な情報が参考情報として LLM に入力されていた可能性が考えられる．一方で，提案手法 ($k = 3$) において Recall がベースラインよりも高いままであるのは，検索結果の網羅性があまり変わらなかったことが考えられる．検索結果の網羅性とは，検索結果において網羅的な回答をするための重要な参考情報が含まれているか否かである．また，提案手法 ($k = 2$) において三つの指標が最も高かったのは，$k = 2$ の時点で検索結果の網羅性が高かった，かつ重複した情報が少なかったことが原因として考えられる．

5.3.2　指示書管理 No.23111000

「指示書管理 No.23111000」における仕上げ工程の立案を行った際の実験結果及び検定結果を表 5.2 に示す．表 5.2 では，三つの指標とも 0 に近い結果となった．提案手法 ($k = 2$) において，Precision が 0.019，F 値が 0.003 向上した．一方で，提案手法 ($k = 1$) と提案手法 ($k = 3$) では三つの指標とも 0 で，全く正解作業を予

表 5.2　5.3.2 節におけるベースラインと提案手法の評価指標比較と p 値

手法	**Precision**	**Recall**	**F 値**	p **値**
ベースライン	0.014	0.009	0.011	-
提案手法 ($k = 1$)	0.000	0.000	0.000	0.343
提案手法 ($k = 2$)	**0.033**	0.009	**0.014**	0.872
提案手法 ($k = 3$)	0.000	0.000	0.000	0.343

表 5.3　5.3.3 節におけるベースラインと提案手法の評価指標比較と p 値

手法	**Precision**	**Recall**	**F 値**	p **値**
ベースライン	0.223	0.089	0.119	-
提案手法 ($k = 1$)	0.317	0.134	0.165	0.284
提案手法 ($k = 2$)	**0.412**	**0.228**	**0.184**	0.213
提案手法 ($k = 3$)	0.345	0.100	0.144	0.543

測することができなかった．対応のある 2 標本 t 検定を行った結果，提案手法三つとも p 値が有意水準 0.05 を上回り，F 値の平均値に有意な差が確認されなかった．

三つの指標とも 0 に近い結果となった原因として，表 5.1 で用いた文書データの文字数が約 3000 文字に比べ，表 5.2 で用いた文書データの文字数が約 300 文字と文章量が非常に少なかったことが挙げられる．従って，仕上げ工程における仕上げ作業名を出力するには，文書データの文章量が必要である可能性がある．

5.3.3　指示書管理 No.23111601

「指示書管理 No.23111601」における仕上げ工程の立案を行った際の実験結果及び検定結果を表 5.3 に示す．表 5.3 では，提案手法を適用したことでベースラインと比べ Precision が最大 0.189，Recall が最大 0.139，F 値が最大 0.065 向上した．表 5.1 と同じく Recall がベースラインよりも向上したことより，仕上げ工程の立案には望ましい結果となった．提案手法 ($k = 2$) において，Precision，Recall，F 値の三つの指標とも最も高かった．しかし，提案手法 ($k = 3$) では，表 5.1 とは異なり，Precision がベースラインよりも高いことが確認された．対応のある 2 標本

t 検定を行った結果，提案手法三つとも p 値が有意水準 0.05 を上回り，F 値の平均値に有意な差が確認されなかった．

我々は提案手法 ($k = 3$) において Precision がベースラインよりも高いままであった原因として，検索結果における重複した情報が表 5.1 の実験よりも LLM の推論に与える影響が少なかったと考えた．表 5.1 の考察と同じく提案手法 ($k = 2$) において三つの指標が最も高かったのは，$k = 2$ の時点で検索結果の網羅性が高かかった，かつ重複した情報が少なかったことが原因として考えられる．表 5.3 における「指示書管理 No.23111601」の文書データの文字数は約 600 文字で，表 5.2 における「指示書管理 No.23111000」の文書データの文字数の 2 倍である．ベースラインにおける三つの指標の数値は 0.3 未満だが，Recall が最大で 0.139 向上したことを確認できるほどの結果であったため，少なくとも文書データの文字数は 600 文字以上が望ましいと考えられる．

第 6 章　おわりに

本研究では，金型製造の仕上げ工程を自動的に生成する RAG の精度向上が目的である．しかし，単一の検索クエリを用いた RAG では，検索結果が参考情報として利用者にとって不十分な場合でもそれを補完する仕組みがなく，検索結果に情報不足が生じてしまう問題点がある．我々はこの問題点を解決するために，LLM の自問自答を用いて検索結果の不足情報を補う手法を提案した．

本手法の有効性を検証するためにベースラインとして単一の検索クエリを用いた RAG との比較実験を行った．本実験では，三つの文書データに対して構築した RAG を用いて，それぞれの文書データに対応した仕上げ工程の立案を行った．提案作業と正解作業を比較し Precision，Recall，F 値を求め，提案手法における F 値の変化に統計的な有意差があるか否かを確認するために対応のある 2 標本 t 検定を行った．

実験の結果，三つの文書データのうち二つの文書データにおいてベースラインと比べ，Recall が向上した．「指示書管理 No.23102415」の文書データでは最大で 0.214，「指示書管理 No.23111601」の文書データでは最大で 0.139 向上した．仕上げ工程の立案する上で工程の網羅性が求められるため，Recall が向上したことについては結果が良かったといえる．実験結果から，Precision，Recall，F 値の三つの指標について，提案手法における不足情報を用いた検索を 2 回行った場合が一番数値が高かった．ここで我々は，不足情報を用いた検索を 2 回行った場合に三つの指標が一番高かった要因として，検索結果の網羅性が高かったことかつ重複した不要な情報が少なかったことだと考えた．しかし，「指示書管理 No.23111000」の文書データではベースラインと提案手法両方とも，三つの評価指標の値が 0 に等しい結果となった．「指示書管理 No.23111000」の文書データでは残りの二つの文書データと比べ，300 文字と文章量が少なかった．従って，利用者の指示に回答するための参考情報が残りの二つの文書データと比べて少なかったことが原因として考えられる．対応のある 2 標本 t 検定を行った結果，「指示書管理 No.23102415」の文書データのみ，提案手法における不足情報を用いた検索を 2 回行った際に p 値が有意水準 0.05 を下回り，F 値の変化に有意な差があった．

今後の展望として，本実験での問題点でもある文章量不足を改善するため，さら

に文章量を増やした文書データでの検証実験を考えている．今回提供していただいた文書データ以外にも仕上げ工程を立案する上で必要なデータを加える必要があるため，それらのデータを用いて再実験をする必要がある．また，本実験では LLM に対してファインチューニングを行わなくても Recall が向上したことが確認された．我々は，ファインチューニングを行った場合に三つの指標がどのように変化するかを確かめるための実験も考えている．さらに，本実験では仕上げ作業の候補だけの出力だったが，我々は仕上げ工程を立案するという目的に対して仕上げ作業の順番を考慮した手法も提案する必要があると考えた．

謝辞

今年１年間研究を進めるにあたり，たくさんの方から支えてくださり本当に感謝しております．まず，鈴木優准教授には，学部 3 年生の仮配属の時からたくさんご指導いただきました．研究で行き詰まった時は時間をかけてアドバイスをしてくださり，まだまだ自分が勉強不足だということに気づきました．この１年間を通して，研究を進める上での計画性や基礎知識の勉強がとても大切だと分かりました．本当にありがとうございました．

事務補佐員の井尾さんには，SA などの続きをする際に大変お世話になりました．研究や学校でのアルバイトを円滑に進めることができたのも井尾さんのおかげだと思っています．たまに雑談をするのが楽しかったです．本当にありがとうございました．

本研究を進めるにあたり，貴重なデータをご提供をいただきました株式会社黒田製作所，株式会社ピュアシステムの皆様に深く感謝申し上げます．打ち合わせでは多くのご助言をいただき，これからの研究の方針についても考える良い機会となりました．今後ともよろしくお願い申し上げます．

研究室の先輩方や同期，後輩には普段の研究でも学校生活以外でもお世話になりました．特に，先輩方には 3 年生の頃から私の研究について相談に乗ってくださりありがとうございました．来年からは院に進学し先輩が 2 人と寂しいですが，とてもとても頼りにしています．これからもよろしくお願いしいたします．また，同期がいたからこそ共にいろんな苦難を乗り越え，なんとかここまでやって来れました．1 年間楽しく過ごせて良かったです．院試の志望理由書で自分の名前を間違える人を初めて見ました．就活の時は特に気をつけてほしいと思っています．後輩達は研究テーマをこれから決めていく時期になると思いますが，頼りになる優しい M2 の先輩が 2 人いるのでたくさん相談に乗ってくれると思います．M1 は全員おそらく自分のことで精一杯になっているかと思いますが，何でも聞いて下さい．

家族には 4 年間の大学生活を過ごしていく上で，生活面や精神面でたくさん支えてくださり感謝してもしきれません．まだ学生生活が 2 年残っているので，頑張ってやりきります．これからもよろしくお願いします．

部活動の先輩や同期，後輩もお世話になりました．最後の 1 年間副主将を務め，

何とか研究と両立ができたことを嬉しく思います．特に，チームを運営する上での統率力や協調性など，チームスポーツから学べるものは全て学べだと思います．

最後に，本稿を仕上げるにあたり支えてくださった方に改めて深く感謝を申し上げます．ありがとうございました．

参考文献

[1] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, Vol. 33, pp. 9459–9474, 2020.

[2] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

[3] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[4] Sunil Arya, David M Mount, Nathan S Netanyahu, Ruth Silverman, and Angela Y Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM (JACM)*, Vol. 45, No. 6, pp. 891–923, 1998.

[5] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, Vol. 35, pp. 24824–24837, 2022.

[6] Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*, 2022.

[7] Yu Wang, Shiwan Zhao, Zhihu Wang, Heyuan Huang, Ming Fan, Yubo Zhang, Zhixing Wang, Haijun Wang, and Ting Liu. Strategic chain-of-thought: Guiding accurate reasoning in llms through strategy elicitation. *arXiv preprint arXiv:2409.03271*, 2024.

[8] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*, 2023.

[9] Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. Corrective retrieval augmented generation. *arXiv preprint arXiv:2401.15884*, 2024.

[10] Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C Park. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. *arXiv preprint arXiv:2403.14403*, 2024.

発表リスト

[1] 藤田将豪，鈴木優『RAG における局所的と全域的な質問の判別』，東海関西データベースワークショップ 2024，2024