IMPERIAL

Imperial College London

Department of Mathematics

PKGE: Macroeconomic Retrieval Augmented Generation with Phased Knowledge Graph-Attuned Embedding

---

Author: Eric Wang (CID: 02437784)

A thesis submitted for the degree of

MSc in Mathematics and Finance, 2023-2024

Declaration

The work contained in this thesis is my own work unless otherwise stated.

Acknowledgements

I would firstly like to thank my industry supervisor from Macro Hive, Dalvir Mandara, who not only proposed the original project topic, but also provided me his expertise and professional insight into LLMs and embedding models. Completing the thesis as part of an internship has been more useful for my career than I ever could have imagined.

I would also like to thank my academic supervisor, Dr. Cristopher Salvi, for his invaluable feedback throughout the project and his support on the more theoretical aspects. He always helped me stay on-track, right until the very end.

Lastly, this project would not nearly have been as enjoyable without sharing the journey with my friends both at Imperial and back in Ireland, as well as my parents. It has been an absolute pleasure.

Abstract

Financial data, particularly macroeconomic data, is notoriously noisy and often difficult to interpret. Large Language Models (LLMs) enhanced with Retrieval Augmented Generation (RAG) offer a powerful and user-friendly interface, capable of answering user queries by accessing a live database. However, the embedding models needed for RAG are pre-trained on large and general datasets, and are typically not optimised for financial and macroeconomic tasks. We propose **PKGE** (**P**hased **K**nowledge **G**raph-attuned **E**mbedding): a commercially viable, multiple-training phase, human-inspired method for fine-tuning embedding models for RAG, using knowledge graphs generated from a domain-specific database of text documents. Our framework enhances embedding training over the baseline vanilla RAG approach, with minimal extra computational complexity, and enables the generation of large amounts of realistic training data from even a very small number of text documents. Our embedding model, PKGE Domain-Specific, demonstrates a notable improvement in empirical retrieval performance on a custom, domain-specific dataset assembled from U.S. Federal Reserve press conference transcripts, as well as excellent real-world results on sample user queries in our demo application.

Contents

1 Architecture 7

1.1 Embedding Models . . . 7

1.1.1 Tokenisation . . . 7

1.1.2 BERT . . . 8

1.1.3 SBERT . . . 8

1.1.4 Sentence Transformers . . . 9

1.1.5 Embedding Dimension and size . . . 10

1.2 The Transformer . . . 10

1.2.1 Attention . . . 10

1.2.2 Encoder & Decoder . . . 10

1.3 The Motivation for PKGE . . . 11

1.3.1 Why Specifically Embeddings? . . . 11

1.3.2 Phased Learning . . . 11

1.3.3 Knowledge Graphs . . . 12

1.3.4 Named Entity Recognition (NER) . . . 13

2 The PKGE Pipeline 15

2.1 General-Domain Embedding . . . 15

2.1.1 Preparing the General-Domain Data . . . 15

2.1.2 Generating the General-Domain Knowledge Graph . . . 16

2.1.3 Graph Post-Processing . . . 18

2.1.4 Generating General-Domain Q&A Pairs . . . 18

2.2 Domain-Specific Embedding . . . 20

2.2.1 Preparing the Domain-Specific Data . . . 20

2.2.2 Data Sources . . . 20

2.2.3 Domain-Specific Q&A Pairs . . . 21

2.2.4 Domain-Specific Test Dataset . . . 21

3 Experiments 22

3.1 Setup . . . 22

3.1.1 Models . . . 22

3.1.2 Datasets . . . 22

3.1.3 Metrics . . . 24

3.2 Fine-Tuning Procedure . . . 24

3.2.1 Training Objective . . . 25

3.2.2 Optimiser and Regularisation . . . 25

3.2.3 Training Parameters . . . 25

3.2.4 Hardware . . . 26

4 Results **27**
4.1 Empirical Results 27
4.1.1 Fed Press Conferences 27
4.1.2 Finance Alpaca 28
4.2 Real-World Testing 28

5 Further Work & Practical Considerations **30**
5.1 Extensions 30
5.1.1 Generalising PKGE 30
5.1.2 Enhancing the RAG Pipeline 30
5.2 Practical Considerations 30

A Prompts, Queries & Answers **33**
A.1 PKGE Application Sample Queries & Responses 33
A.2 PKGE Pipeline Prompts 35

B Definitions **36**

Bibliography **39**

List of Figures

1.1 SBERT Training 8
1.2 SBERT Inference 8
1.3 The Transformer architecture [42]. 11
1.4 A simple knowledge graph example. 13

2.1 The training phase of PKGE, where the embeddings for general domain knowledge are learned. This yields a 'generally-trained' embedding model which can be further fine-tuned on domain-specific data. 15
2.2 The RAG phase of PKGE, where domain-specific documents are provided for fine-tuning. This yields the final fine-tuned embedding model for RAG tasks. 16
2.3 LLM system prompt to generate a list of JSON objects describing the graph edges. 17
2.4 The full general-domain knowledge graph. 18
2.5 A snippet of the general-domain knowledge graph. 18
2.6 LLM prompt to generate Q&A pairs from a given subgraph of nodes and edges. 19

A.1 Sample LLM response on user query with retrieved context: Fed trade ideas. 33
A.2 Sample LLM response on user query with retrieved context: Fed view on inflation. 33
A.3 Sample LLM response on user query with retrieved context: Labour market impact on Fed decisions. 34
A.4 Sample LLM response on user query with retrieved context: Vague query. . 34
A.5 Cleanup prompt for pre-processing press conference PDFs. 35
A.6 LLM system prompt for final output to the user. 35

List of Tables

1.1 Our ontology and category descriptions. 14

3.1 The embedding models we use for our experiments. 23
3.2 Our datasets used. 23

4.1 Results on the Fed press conferences dataset. The top-performing model is highlighted in bold. 27
4.2 Results on the Finance Alpaca dataset. The top-performing model is highlighted in bold. 28

Introduction

To overcome the hallucination (inaccurate or factually incorrect information) often seen in LLM responses, the model needs access to an up-to-date database to retrieve relevant context information for a user's query. This method, proposed in 2020 by Lewis et al., is known as Retrieval Augmented Generation [20]. An LLM enhanced with RAG can act as a search engine interface for the database, providing convenient and intuitive use while being able to support its answers with sources for the information used, if needed.

For financial data, this technique is particularly compelling. Since inaccurate responses from an LLM regarding a trade idea or investment decision may have catastrophic consequences, minimising this risk is crucial. Much of the work around LLMs and embedding models for finance revolves around general financial news, financial statements, and equities-related data, as seen in [2, 49, 48, 50]. We focus specifically on offering insights into macroeconomic topics, such as interest rates, FX, and commodities. For this paper, we restrict the scope to texts and research related to the U.S. Federal Reserve to simplify the task and reduce the amount of data needed.

Our goal is to fine-tune an **embedding model** to retrieve the most relevant context for every query. Embeddings are vector representations of data, which is text in our case, but can also be images and audio [3], for example. By representing text as vectors, we can compare them using similarity metrics such as cosine similarity or Euclidean distance (defined in Appendix B).

With a notion of similarity defined, we can compare the embeddings of different texts, which is crucial for tasks such as question answering and semantic similarity [20]. These will be used in retrieving the most relevant documents from the database by finding the top K closest embedded texts, in cosine similarity, to that of the query.

We generate custom datasets for fine-tuning by creatively applying knowledge graphs, which contain structured information about macroeconomic concepts and entities, such as the interest rate, inflation and the Federal Reserve. An LLM first extracts and distills data from our source documents before generating the graph structure. This then forms the basis for generating question and answer pairs, which is also done via LLM. Our technique also comes with the benefit of being able to generate many new realistic data points from a small corpus. The fine-tuned models are then evaluated on our custom Fed press conferences dataset, as well as a standard finance Q&A dataset.

Our vision for the proposed final system is a commercial, scalable and comprehensive application capable of offering precise, up-to-date domain-specific advice. We distinguish our solution from similar recent works such as FinGPT [49] and FinBERT [2]. Firstly, we concentrate on macroeconomic data, which lacks much representation in financial datasets. Next, we only train an embedding model, rather than a model such as GPT-4o [30] or Google's T5 [27], due to the much lower computational cost and data required. Lastly, we focus on the embedding model and its role as the retriever in the RAG procedure while making use of LLMs in intermediate steps, as well as in generating the final output to the user. At the end, we also discuss potential future work and considerations for practical implementations of PKGE.

Chapter 1

Architecture

1.1 Embedding Models

Text embedding models employ a deep neural network to represent words or sentences [3]. Most importantly, these allow texts to be compared for similarity. Embeddings also allow for visual representations through techniques such as t-SNE [41], which can aid data analysis by depicting clusters.

Within deep neural network architectures, embedding layers are able to encode representations of text features, which helps the model to generalise to unseen data. Embeddings can also “map categorical inputs to continuous vectors, facilitating backpropagation and optimisation” [3].

Our purpose for embeddings is semantic search. By learning the association between the semantic meanings of words, such as common word pairings and synonyms, embedding models are able to determine similarity between two text documents accurately, even without the same key words appearing in both texts. For example, a macroeconomic-specific embedding model should be able to make a close association between “monetary policy” and “interest rate”, or indeed between “interest” and “rate”. Mathematically, we mean that the embedding vectors generated from the two similar texts should have a cosine similarity close to 1. A model which is not sufficiently trained on financial language may also mistake the word “interest” for meaning curiosity or intrigue instead.

With a macroeconomic-specific embedding model, we hope to improve the accuracy of Retrieval Augmented Generation for answering user queries. By retrieving relevant and precise context information from a database, an LLM can quote information sources in its response, which helps to reduce hallucination. Without such context, an LLM may incorrectly interpret its training data, or produce responses which follow unexpected patterns in the data [15], resulting in nonsensical or factually incorrect answers.

1.1.1 Tokenisation

To be inputted into the model, text is first **tokenised**. It is separated into smaller words or fragments, which are not necessarily words themselves. One such method is called WordPiece [9], which is the tokenisation method originally used by Google in the the BERT model [7]. This is also used in the Sentence Transformer models [37] which we implement in this project.

Tokens allow a model’s entire “vocabulary” to be represented using smaller fragments of words, which are then combined together to form the original word. This allows any alphanumeric to be tokenised, and ensures an embedding can always be generated. Condensing the vocabulary into word pieces is also much more efficient on storage.

1.1.2 BERT

BERT is a pioneering encoder-only architecture developed in 2019. The model consists of several layers of encoder models trained to learn representations of a sequence of input tokens for purposes such as semantic similarity, or comparing meanings of different pieces of text [7].

Encoder-only architectures seek to understand texts beneath the surface, such as their sentiment and semantic meaning. They take text as input and output a vector, which is the embedding we want. BERT's special [CLS] token allows it to perform classification tasks, while its bidirectional attention allows it to see forwards and backwards [19], meaning that it can learn from the entire input at once and understand the context of every token. This is different from the architecture of decoder models such as GPT-4o [30], which can only see the remaining tokens after each token in the input, allowing it to perform well on tasks such as logical continuations of input text.

However, traditional embedding models such as BERT and Word2Vec [24] work only on individual words, and are not effective for use on sentences or entire paragraphs. We now introduce SBERT, which remedies this.

1.1.3 SBERT

Sentence BERT, or SBERT, addresses the original model's inefficient comparison of sentence embeddings. This is needed for tasks such as semantic search, for example [37]. BERT does not compute independent sentence embeddings, and methods such as averaging word embeddings to create a fixed-length embedding for a sentence did not demonstrate useful results.

A disadvantage of the original BERT architecture is that when two sentences are input, the embeddings that are computed are relative to each other, and cannot be used for each sentence independently for comparison against other sentences. This problem is solved by SBERT [8].

SBERT allows sentences and longer text chunks to be compared using cosine similarity, while offering a significant speedup in time to compare embeddings. The architecture for fine-tuning is shown in Figure 1.1, while Figure 1.2 depicts the architecture at inference.

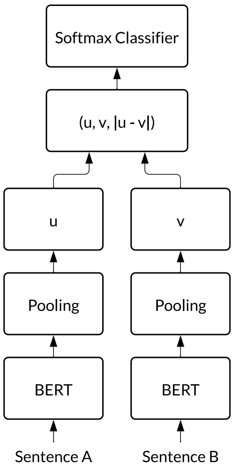


Figure 1.1: SBERT Training

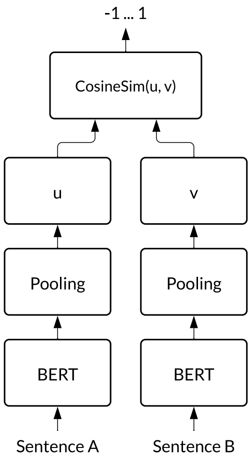


Figure 1.2: SBERT Inference

During training, Sentence A is considered as the 'anchor', while Sentence B is the 'positive'. The BERT networks are siamese, meaning that their weights are linked [33]. Essentially, this results in only one model being trained, since the weights are identical, but the two sentences are still processed separately. Sentence A is embedded first and is then followed by Sentence B.

Each BERT produces 512 768-dimensional embeddings, which are then averaged using mean pooling to convert the original 512×768 matrix to 1×768, while ignoring padding values. This procedure can be parallelised over several batches of sentence pairs [33].

1.1.4 Sentence Transformers

The Sentence Transformers Python library [37] offers a variety of capable pre-trained SBERT embedding models which are designed for sentences and longer texts, and are also easy to fine-tune.

Fine-tuning of a *SentenceTransformer* embedding model is done by minimising the Multiple Negatives Ranking Loss, defined in the Sentence Transformers documentation [37]. To preface the definition, we introduce some preliminaries.

Definition 1.1.1. (Positive & Negative Pairs) *Consider a corpus with* N *queries* $\{q_i\}_{i=1,\dots,N}$ *and* N *answers* $\{a_j\}_{j=1,\dots,N}$. *A query and answer pair* (q_i, a_j) *is called a positive pair if* $i = j$ *and a negative pair if* $i \neq j$ *i.e. for each query* q_i*, we define exactly* 1 *correct answer and* $N - 1$ *incorrect/irrelevant answers.*

Definition 1.1.2. (Binary Cross Entropy) [45] *Let* y *be the true value of an observation, and let* $\hat{y}$ *be the predicted value. The binary cross entropy loss is given by*

$$-y \log(\hat{y}) - (1 - y) \log(1 - \hat{y}).$$

Definition 1.1.3. (Softmax Activation Function) [4] *Let* z *be the vector of logits produced by the embedding model on answers* $\{a_j\}_{j=1,\dots,N}$ *to a query* q_i*, where* z_i *is the i-th element of* z. *The softmax activation function is given by*

$$\frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}}.$$

The softmax function transforms the vector of logits, which represents relative scores for the relevance of each paragraph to the query, to a vector of probabilities. That is, the softmax function maps values in $\mathbb{R}$ to $[0, 1]$, with the sum of all the probabilities equal to 1. This allows the most relevant answers to be ranked [4].

Now, we can discuss the Multiple Negatives Ranking Loss. We define the loss for each batch of Q&A pairs as follows:

Definition 1.1.4. [33] (Multiple Negatives Ranking Loss) *For each batch with* b *pairs, we begin by computing the cosine similarities between every possible query and answer pair. Let* B *denote the matrix of cosine similarities, i.e.*

$$B = \begin{pmatrix} \mathrm{CosSim}(q_0, a_0) & \cdots & \mathrm{CosSim}(q_0, a_b) \\ \vdots & \ddots & \vdots \\ \mathrm{CosSim}(q_b, a_0) & \cdots & \mathrm{CosSim}(q_b, a_b) \end{pmatrix}.$$

Next, define $I := I_b$ *to be the* $b \times b$ *identity matrix. This represents the ideal cosine similarities. Then, for each* $B_{i,j}$ *and* $I_{i,j}$*, we aim to minimise the binary cross entropy loss between the computed cosine similarity and ideal class (0 for a negative pair and 1 for a positive pair):*

$$\sum_{i=1}^{b} \sum_{j=1}^{b} -I_{i,j} \log(B_{i,j}) - (1 - I_{i,j}) \log(1 - B_{i,j}). \tag{1.1.1}$$

This is the Multiple Negatives Ranking Loss for each batch, and can be summed over all batches to obtain the overall loss.

Splitting the corpus into batches allows to take advantage of parallel computing on a GPU. Larger batches come with the benefit of allowing the Multiple Negatives Ranking Loss to be calculated on more Q&A pairs at once, which generally increases performance [37]. Incidentally, larger batch sizes have higher VRAM requirements when implementing on a GPU. In our experiments, computation time increased non-linearly as the maximum VRAM capacity was approached through increasing batch sizes. It is therefore vital to find a good balance between batch size and training time.

1.1.5 Embedding Dimension and size

Embedding vectors are often several hundred dimensions, but this can be adjusted to adapt to the amount of data available. Typical sizes include powers of two and related numbers, such as 1024, 768 and 384. Larger embedding dimensions lead to larger models sizes and computational complexity, but a greater ability to capture features in its data [29]. We select a model with dimension 768 since it balances training and inference speed with sufficient capability in understanding our domain.

Additionally, Sentence Transformer embeddings are pre-trained to produce vectors of length 1. We do not consider the length of the embedding vector for our purposes, so it is beneficial to have normalised vectors. Additionally, this means that when calculating similarity, the dot product would yield the same interpretation as the cosine similarity.

1.2 The Transformer

1.2.1 Attention

The foundation of contemporary LLM architectures is the Transformer, pioneered by Google in 2017 [42]. Crucially, this design introduced the concept of **attention**, which allowed language models to understand context and relationships between words. **Self-attention** refers to how closely an input word is related to other words within its own context, while **encoder-decoder attention** keeps track of the significance between words across the encoder's and the decoder's representations. Encoder-decoder attention ensures that the meaning of important words in an input are not lost, which is particularly useful for translation or instruction-following tasks [43].

1.2.2 Encoder & Decoder

We consider the encoder-only architecture, and do not worry about fine-tuning the weights of the decoder. The encoder architecture is designed to learn effective representations of data as vectors, and is different from decoders in that they can see forwards and backwards from each token [19]. Decoders can only see forwards from each token, resembling the way a human would continue a partially complete sentence. This allows encoders to fully understand the semantic meaning of their input, rather than being good at predicting the most reasonable continuation to a given text.

Finally, encoder-decoders may be particularly compelling for a future extension of this project. These implement the full Transformer architecture, and allow information from the encoder to be passed to the decoder, which has the potential to significantly boost our framework's capability. Currently, PKGE only implements a pre-trained decoder-only LLM at the output step to save on the computation required to fine-tune a full encoder-decoder model. Google's T5 [27] offers excellent pre-trained models which may be used for this purpose, but this is unfortunately beyond the scope of this project. Even with such a model at the output stage, we still need a well-trained retriever to fetch relevant context information. We continue to demonstrate our framework's effectiveness on fine-tuning the

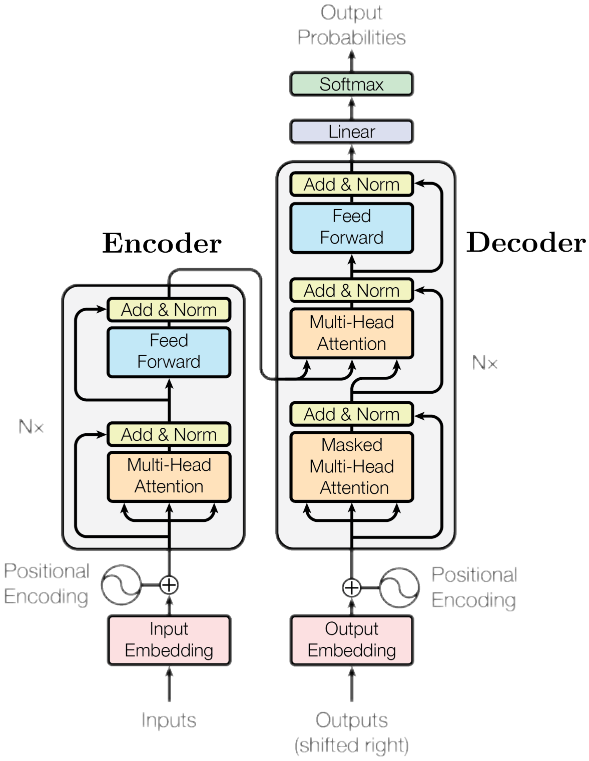


Figure 1.3: The Transformer architecture [42].

embedding model specifically for accurately retrieving macroeconomic data to answer user queries.

1.3 The Motivation for PKGE

1.3.1 Why Specifically Embeddings?

Given our focus on commercial applications, we want to minimise the compute required to produce, implement and run our framework. Firstly, we avoid pre-training entirely. The pre-training of any model requires a very large amount of data and prohibitive GPU compute costs. Therefore, we build our model on the foundation of already pre-trained models, and fine-tune on top of those. This is much more viable and still shows excellent results.

We target the embedding model for improvement as its architecture has relatively few parameters compared to others. Our goal is to improve its semantic understanding of macroeconomic language and to retrieve the most relevant context information for user queries relating to the U.S. Federal Reserve, which is the area on which we demonstrate our results in this paper. By restricting our scope, we reduce the computational effort and the amount of data needed. It is significantly easier to fine-tune an embedding model with several hundred million parameters, than even small LLMs such as Llama3-8B, with 8 billion parameters [1].

1.3.2 Phased Learning

The most common method for fine-tuning LLMs on domain-specific data today involves directly incorporating the data into the pre-trained model and adjusting the model's weights according to an optimisation procedure like Adam [18], such as in [31]. We believe that this direct fine-tuning method does not provide the model with the necessary foundational knowledge to learn the most effective embeddings for the data.

IBM demonstrated the efficacy of a simple-to-complex task training procedure for LLMs, improving scalability as well as performance on specific tasks[38]. They first trained the model on simple tasks before moving to more complex ones, and illustrated that training on both kinds at the same time lowered performance. Our procedure involves progressively narrowing the scope of the training data, rather than increasing complexity in the questions and answers. The domain contains mostly objective facts and theory, which results in a broad range of knowledge. Accordingly, the model would need a large quantity of data and exposure to the entire spectrum for a full understanding. Again, we demonstrate our framework using only data relating to the Fed to reduce the amount of documents required.

By "warming up" the embedding weights to the general context of macroeconomic theory, as well as relevant articles, documents and reports, we hope to point the model in the right direction prior to domain-specific fine-tuning.

The "phased" part of the PKGE method refers to how we approach the complexity of the information that the model learns. Making use of transfer learning [25], we begin with a pre-trained model. This sets a good foundation and ensures it can be trained for our tasks with well-engineered prompts. We take inspiration from the way humans learn and gain experience in specific fields; Firstly, the embedding model is "generally-trained" on a large quantity of academic data in one subject, simulating a university undergraduate or master's level education. We ensure that the data used is diverse, covering a wide range of topics and question types. This simulates the way humans would think when answering questions in the LLM's position.

The domain-specific fine-tuning phase represents career experience, domain-specialisation, or possibly PhD-level knowledge. The motivation is that, with a strong foundation in the relevant general knowledge area, the embedding model is able to more easily and efficiently learn specific topics within that area, just like in humans. In Chapter 4, we show that, although our empirical results are mixed, phased learning has potential to add value given a larger quantity of available training data.

1.3.3 Knowledge Graphs

A knowledge graph (KG) is defined as a network of real-world entities, such as objects, events, situations or concepts, which illustrates the relationships between them [16]. Typically, this graph is directed, and edges are taken from a pre-defined list of relation types. An example of a real-world implementation is Google's knowledge graph, used to display a panel of relevant information relating to a search query beside other search results [11].

We adopt a more general approach to the knowledge graph concept. By adapting the *graph_maker* library for Python [28], which uses a pre-trained LLM to create graph edges from a set of text documents, we can easily create the KGs we need for embedding training. In Section 2.1.1, we further discuss how we apply KGs in the training of embedding models.

The key differences in our approach are the lack of directed edges, as well as a unique, LLM-generated description for each graph edge. This is because we are using the knowledge graph in a different way. Firstly, we use KG generation as a method of distilling the information contained in the original documents. By asking the LLM to extract only the entities and their relationships from the text, we make it easier to generate meaningful Q&A pairs later in the pipeline, and at the same time maintain a straightforward way to restore the data to text form using the LLM again. Our approach combines the nodes and edge descriptions with each original chunk of text from which they were extracted to enrich the LLM's understanding at the output stage, when the retrieved context is used to generate the LLM's final response.

Importantly, we also avoid pre-defining a list of relations and avoid a directed graph. In the macroeconomic context, both of these may be very difficult to apply accurately. It is already challenging for a human to label relations between economic quantities and entities due to their noisy and unpredictable interactions, and there is no accurate data which the embedding model can be trained on. As a consequence of inaccurate data, the embedding model may learn misleading or inconsistent relations.

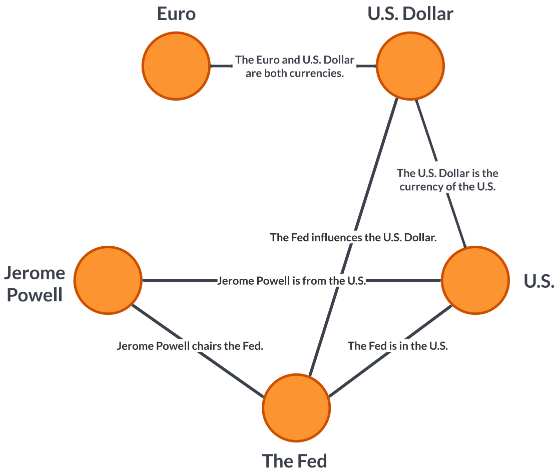

Euro
U.S. Dollar
The Euro and U.S. Dollar are both currencies.
The U.S. Dollar is the currency of the U.S.
The Fed influences the U.S. Dollar.
Jerome Powell
Jerome Powell is from the U.S.
U.S.
Jerome Powell chairs the Fed.
The Fed is in the U.S.
The Fed

Figure 1.4: A simple knowledge graph example.

The effectiveness of the knowledge graph structure becomes especially apparent for the domain-specific documents database. It can be easily scaled as new documents arrive; One only needs to determine the relationship between new concepts and old ones, and adjustments/insertions can be easily made. Additionally, metadata can be added to each edge to make this process easier, such as the article title, date published and any information which may affect previous data, such as a change in view or a closed trade. Storing edges in a Python list-like object also allows for easy updating. In production, however, one should consider a graph database for greater storage capacity and data security.

Moreover, the structure allows for the creation of subgraphs, linking several related concepts together. Since we can create a relatively large number of subgraphs from our main graph, this has the potential to generate many new high-quality data points through an LLM. Since we always have the context information available, we minimise hallucination and can create many diverse question and answer pairs using unique combinations of graph edges. This is particularly effective for our small set of domain-specific documents, turning around 100 text chunks into almost 3,000 through taking subgraphs of various sizes. More about this procedure is discussed in Section 3.1.2.

Finally, a natural visualisation of the way the human brain stores information would be a connected graph of concepts, ideas and memories. This is exactly what we aim to recreate with PKGE; Conceptually, we aim to mimic a human understanding of our topic. A simple example of our knowledge graph structure is depicted in Figure 1.4, which shows how a basic graph of Fed-related concepts may be connected and described.

1.3.4 Named Entity Recognition (NER)

Named Entity Recognition is a natural language processing technique which recognises pre-defined categories of objects in text [14]. This is especially important when it comes to macroeconomic texts, since there is a wide range of commonly used names which refer to the same entity or concept. For example, the U.S. Federal Reserve may also be called ‘the Fed,’ or the ‘Federal Reserve System.’

An **ontology** provides labelled categories of the entities to be extracted to the LLM, identifying the entire range of entities to be expected in a text. The labels, as well as the descriptions we have used, are presented in Table 1.1.

Category	**Description**
Economic Quantity	An important quantity for those who study economics, such as economists, researchers and governments.
Person	Singular person name without any adjectives. Remember a person may be referenced by their name or using a pronoun.
Concept	A noun relating to an academic definition or idea. This should NOT include verbs, times, years and historical events.
Place	A physical location. This should NOT include times, years and historical events.
Action	The fact or process of doing something, typically to achieve an aim, e.g. increase, decrease, control, prevent, influence.
Document	A piece of written, printed, or electronic matter that provides information or evidence or that serves as an official record.
Organisation	A governing body or group of individuals who have a shared goal.
Event	A situation or occurrence involving multiple people. Do not include qualifiers or verbs like gives, leaves, works etc. This should NOT include academic concepts or definitions. This should NOT include times, years and historical events.
Miscellaneous	Any important concept can not be categorised with any other given label. This should NOT include times, years and historical events.

Table 1.1: Our ontology and category descriptions.

General-purpose NLP libraries such as *NLTK* [5] and *spaCy* [12] have shown inconsistent results for categorisation according to our ontology. These require significant manual labelling to work well. In testing these libraries, they empirically demonstrated poor entity resolution performance due to the nuances of our domain. Often, they are unable to even distinguish between different names for the Fed. Therefore, we consider alternatives.

Rather than using deep learning methods to resolve named entities, it turned out that a human-defined query table was most effective. Given the diverse expressions used to refer to macroeconomic concepts and organisations, we feel that manual labels that are verified by domain experts would achieve the best results, especially given that this process only needs to be done once. There is potential to expand the list to cover a wider range of concepts, though it would become less practical. Future work may involve searching for a more scalable solution to this problem.

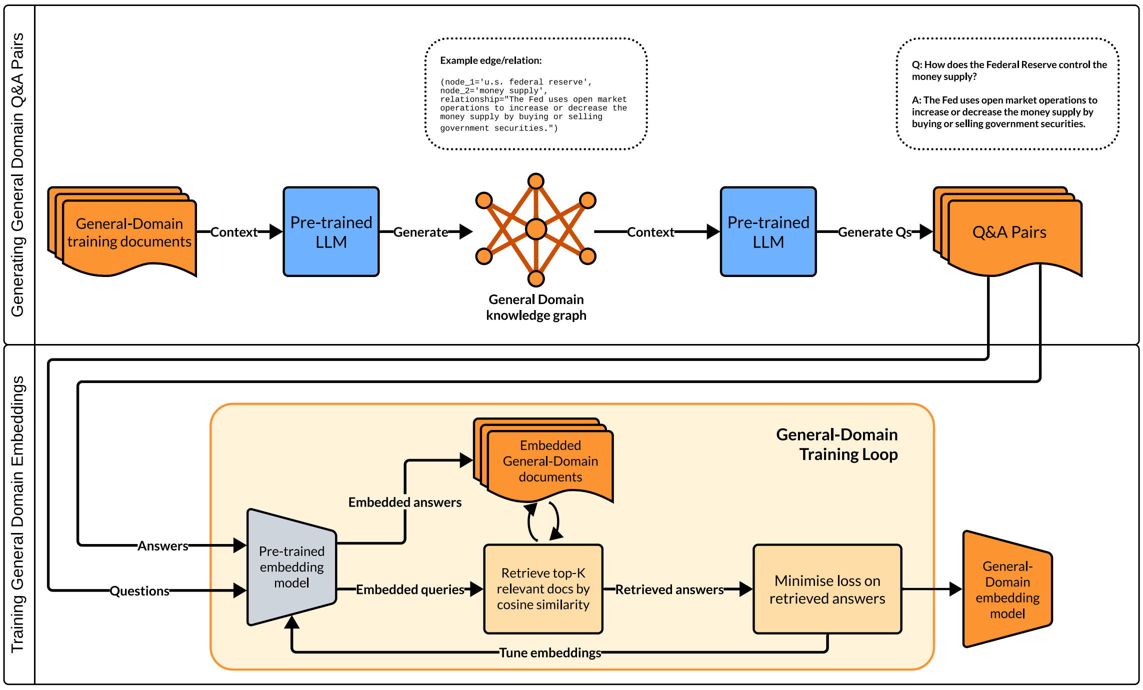
The resolved entity names can then be applied in the graph, which makes it more densely connected. This is because two entities which are not resolved may have their own edges with other nodes, and may not necessarily be connected themselves. This is an artifact of graph generation using LLMs, and more on the procedure will be explained in Section 2.1.2. Unifying all entities referring to the same real-world concept allows that entity's node to have many connections and aids in Q&A generation further down the pipeline, when we need to determine which other entities it would be related to.

Chapter 2

The PKGE Pipeline

PKGE, pronounced *package*, is split into two phases: The first involves learning general knowledge about a particular domain, which is macroeconomics in our context, while the second involves fine-tuning on specific areas within the general domain, which relate to the U.S. Federal Reserve in this case.

Figures 2.1 and 2.2 illustrate our proposed framework.



Example edge/relation:

```
(node_1='u.s. federal reserve',
node_2='money supply',
relationship="The Fed uses open market
operations to increase or decrease the
money supply by buying or selling
government securities.")
```

Q: How does the Federal Reserve control the money supply?

A: The Fed uses open market operations to increase or decrease the money supply by buying or selling government securities.

Generating General Domain Q&A Pairs
General-Domain training documents
Context
Pre-trained LLM
Generate
General Domain knowledge graph
Context
Pre-trained LLM
Generate Qs
Q&A Pairs
Training General Domain Embeddings
General-Domain Training Loop
Embedded General-Domain documents
Embedded answers
Answers
Questions
Pre-trained embedding model
Embedded queries
Retrieve top-K relevant docs by cosine similarity
Retrieved answers
Minimise loss on retrieved answers
General-Domain embedding model
Tune embeddings

Figure 2.1: The training phase of PKGE, where the embeddings for general domain knowledge are learned. This yields a 'generally-trained' embedding model which can be further fine-tuned on domain-specific data.

2.1 General-Domain Embedding

2.1.1 Preparing the General-Domain Data

Since embedding models for RAG applications play the role of the retriever, we need a method of tuning them such that they are best prepared to deal with user queries requiring retrieval. A question and answer objective has been shown to be effective in preparing an embedding model for such tasks [17].

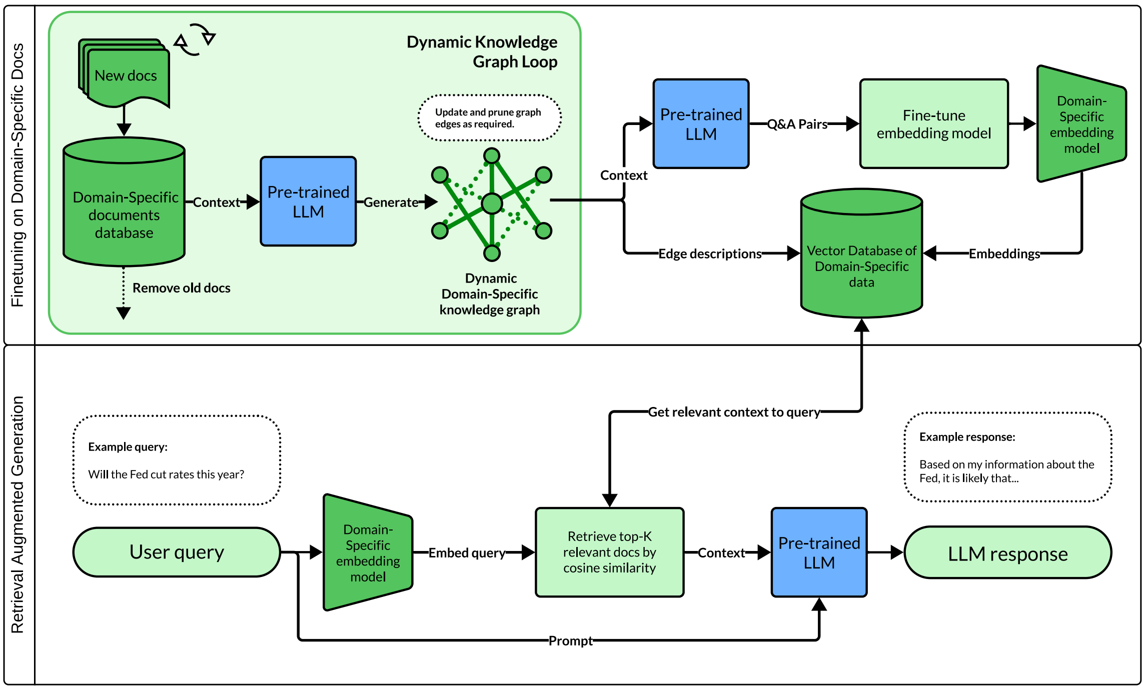

Finetuning on Domain-Specific Docs
New docs
Dynamic Knowledge Graph Loop
Update and prune graph edges as required.
Domain-Specific documents database
Context
Pre-trained LLM
Generate
Dynamic Domain-Specific knowledge graph
Remove old docs
Context
Pre-trained LLM
Q&A Pairs
Fine-tune embedding model
Domain-Specific embedding model
Edge descriptions
Vector Database of Domain-Specific data
Embeddings
Retrieval Augmented Generation
Get relevant context to query
Example query:
Will the Fed cut rates this year?
Example response:
Based on my information about the Fed, it is likely that...
User query
Domain-Specific embedding model
Embed query
Retrieve top-K relevant docs by cosine similarity
Context
Pre-trained LLM
LLM response
Prompt

Figure 2.2: The RAG phase of PKGE, where domain-specific documents are provided for fine-tuning. This yields the final fine-tuned embedding model for RAG tasks.

Of course, in order to generate the Q&A pairs, we first gather a large collection of general-domain text documents. Topics include, but are not limited to:

- Macro and microeconomic theory
- The interplay and relationship between different economic quantities
- The U.S. Federal Reserve
- How decisions by the Fed affect the market
- Monetary and fiscal policy
- Asset classes and how they are affected by the Fed's decisions

The data is pre-processed slightly by hand. Articles are split into paragraphs of text, each of which is considered as a separate "document". Importantly, this means that no information is shared between paragraphs of a document, from the LLM's perspective. Therefore, input texts should be written in such a way that paragraphs are mostly self contained for best results, though this is not essential. We define a dictionary of common variations of macroeconomic concepts and entities so that our procedure is able to recognise variants and form dense connections between concepts, rather than splitting them among different variants of the same entity.

By training the embeddings to become familiar with our domain, the model is guided in the right direction for further fine-tuning in the later phase.

2.1.2 Generating the General-Domain Knowledge Graph

Now, with our training data, we make use of a pre-trained LLM to generate the edges of the knowledge graph. In 2019, OpenAI demonstrated that LLMs are capable of tasks such as reading comprehension, translation, summarisation and question answering with zero-shot prompts [34].

Naturally, we make use of this ability. We employ both OpenAI's GPT-4o [30] along with an adapted version of the GraphMaker library [28] to generate variations of the knowledge graph using a finely engineered prompt. This model is the current state-of-the-art for all-purpose tasks, though we have also tested Llama 3-70B and observed acceptable results. The details of the prompt are shown in Figure 2.3.

> You are a network graph maker who extracts terms and their relations from a given context.
> You are provided with a context chunk (delimited by ```). Your task is to extract the ontology of terms mentioned in the given context. These terms should represent the key concepts as per the context.
> **Thought 1:** While traversing through each sentence, think about the key terms mentioned in it.
>
> - Terms may include object, entity, location, organization, person, condition, acronym, documents, service, concept, etc.
>
> **Thought 2:** Think about how these terms can have a one on one relation with other terms.
>
> - Terms that are mentioned in the same sentence or the same paragraph are typically related to each other.
> - Terms can be related to many other terms, so maximise dense relationships between terms when possible.
>
> **Thought 3:** Find out the relation between each such related pair of terms and try to minimise the number of connections needed.
> **Thought 4:** You want to produce a very well connected and dense graph, with as much overlap as possible.
> **Thought 5:** Remember that some phrases and names have many variations, but they all refer to the same entity, so make sure to be consistent in their naming.
> Format your output as a list of JSON. Each element of the list contains a pair of terms and the relation between them, like the following:
>
> ```
> [
>   {
>     “node_1”: “A concept from extracted ontology”,
>     “node_2”: “The related concept from the extracted ontology”,
>     “edge”: “relationship between the two concepts, node_1 and node_2 in
>     one or two sentences”
>   },{...}
> ]
> ```

Figure 2.3: LLM system prompt to generate a list of JSON objects describing the graph edges.

With the edges generated, we can now visualise the dense network produced by the LLM. This is depicted in Figure 2.4.

The graph has over 3,000 edges, with hotspots representing particularly well-connected nodes, such as ‘Inflation’, ‘U.S. Federal Reserve’ and ‘Interest Rate’. Since these kinds of topics are discussed in greater detail and across more documents in the corpus, more connections are created by the LLM. A zoomed-in snippet of the graph is shown in Figure 2.5, which depicts some of the nodes. Accordingly, less common concepts will not be as well connected, which can be seen around the edges in Figure 2.5.

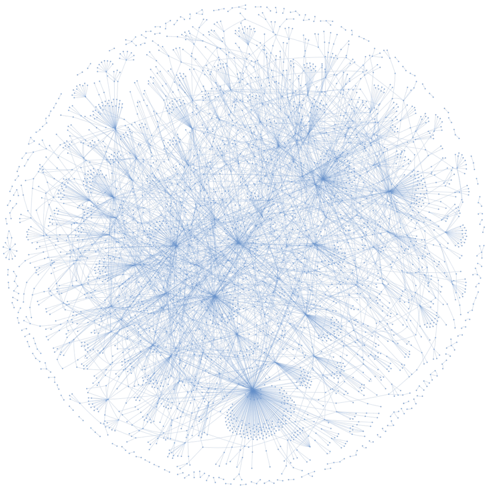


Figure 2.4: The full general-domain knowledge graph.

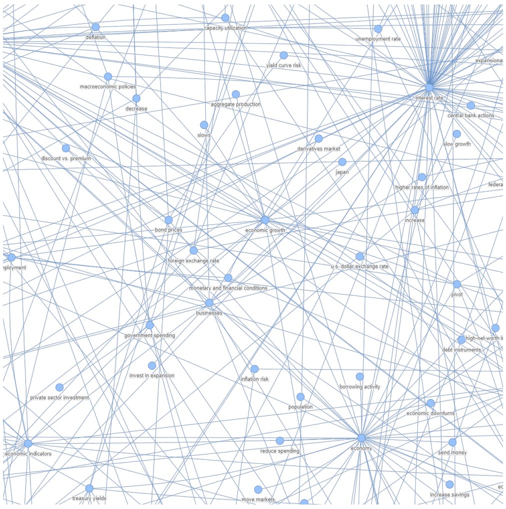


Figure 2.5: A snippet of the general-domain knowledge graph.

Since different documents may refer to these concepts in different ways, entity resolution minimises stray nodes which appear unconnected, yet refer to the same concept. Since our NER method is not all-encompassing, a number of unconnected smaller graphs remain. These generally include more obscure topics, poorly-phrased node names, or otherwise bad quality extractions from the LLM. We now discuss post-processing the generated graph to resolve these issues.

2.1.3 Graph Post-Processing

Before generating the Q&A pairs, we first prune the graph to only the top 1,000 nodes of highest degree (nodes which have the most edges, including loops). This eliminates many of the smaller subgraphs on the outside which are not connected to the "main" graph (see the edges of Figure 2.4). Importantly, the LLM sometimes extracts nodes which are uninformative, unusually phrased or otherwise not useful. These comprise most of the less well connected nodes and subgraphs. Pruning may remove some useful information, but mostly has the effect of reducing anomalies.

Additionally, the LLM sometimes generates "multi-edges." This means that the same two nodes can have more than one edge between them, again due to similar concepts and relations being mentioned across documents. Resolving this is straightforward; Each group of multi-edge descriptions is collected and fed into another LLM prompt in order to summarise them in a few sentences. The extra edges are removed, and a single, unified edge replaces them.

With post-processing completed, the graph is now ready for generating the Q&A pairs.

2.1.4 Generating General-Domain Q&A Pairs

Our model is designed for long-form question answering (LFQA), inspired by the work in [6]. This objective involves the understanding that many short queries require answers which are often several sentences to a paragraph long. We want the embedding model to associate these shorter queries with longer answers to hopefully improve retrieval on large paragraphs. The authors of the paper introduce the *FinTextQA* dataset for this purpose, specific to the financial domain. Although this dataset would have been ideal for training or evaluating PKGE, it has not yet been made public at this moment. Instead, we create our own macro-specific LFQA datasets using our knowledge graph.

By generating question and answer pairs based on pairs of nodes, as well as subgraphs of three and four nodes, we are able to capture more complex, multi-hop connections between concepts, which the embedding model can learn from. One of the most important features of the PKGE framework is its ability to generate a large number of high-quality Q&A pairs from a small number of source documents. This is done by exploiting subgraphs to create unique groupings of nodes. Each grouping can then be fed into an LLM prompt to generate the question and answer pairs.

The nodes and edges in each subgraph are first formatted and fed into the LLM prompt as context. The answer generated by the LLM is effectively a summary of the relationship between the nodes, enriched with the context of the original text chunk which yielded the edge, while the question is form-fitted to work with the answer. Other data such as the title and date of the source document may also be stored in each edge's metadata.

Figure 2.6 shows the LLM prompt for generating Q&A pairs from a given subgraph. Note that we make use of few-shot prompting to give the LLM some guidance on what we expect it to produce, which generally improves its ability to learn instructions for a new task [34]. These examples are written by hand and use information available on the Federal Reserve website.

> You are given some context information.
> Your task is to write one very general and interesting question relating directly to the given nodes, relationship and context.
> Remember that there can be several answers to a question, so make sure to add this detail to your answers.
> The question should focus on the nodes provided and their relationship, but expand and diversify your answer using the context and relationship.
> The answer should be at least a few sentences long, but use only the information given, no external knowledge.
>
> Here are some examples of good Q and A pairs:
> Q: How does the Federal Reserve work?
> A: The Federal Reserve conducts the nation's monetary policy to... (truncated)
> Q: What is monetary policy?
> A: The term 'monetary policy' refers to... (truncated)
> Q: What quantities are influenced by the Federal Funds Rate?
> A: The Federal Funds Rate is the interest rate at which... (truncated)
>
> Do not respond with anything else except the Q and A pair.
> The information is below.
> Nodes: {nodes}, Relationships: {edges}, Context: {original_text}
>
> Q: ___
> A: ___

Figure 2.6: LLM prompt to generate Q&A pairs from a given subgraph of nodes and edges.

Capturing all possible subgraphs from our 1,000-node pruned knowledge graph is unnecessary, so we restrict it to several categories: single-edge, single-hop and multi-hop. Single-edge involves two nodes and the edge between them, single-hop involves three connected nodes and two edges, and multi-hop involves four connected nodes and their three edges. We randomly sample a set number of subgraphs from each category: 1,000 single-edges, 2,000 single-hops and 2,000 multi-hops to yield 5,000 Q&A pairs in total. These proportions were chosen to emphasise LFQA. We ensure that the groups are unique: no category has two subgraphs with identical nodes.

All of our generated Q&A pairs are used for model training and validation, but are not used in any evaluation. This is left to external datasets.

2.2 Domain-Specific Embedding

2.2.1 Preparing the Domain-Specific Data

Following our procedure, we now proceed with the fine-tuning phase using live domain-specific data, where the temporal aspect now comes into play. Previously, the general-domain knowledge, which encompasses facts and theories related to financial markets and macroeconomics, had little temporal importance. This is because the data remains largely relevant over time, except during significant events like a change in the Chair of the Fed. With the domain-specific data, the temporal aspect is almost essential. News which affect macroeconomic quantities such as bond yields, FX and commodity prices quickly become stale, so it is crucial that the knowledge graph of these documents are regularly updated and pruned as news arrives and becomes outdated.

By including the date of the document in each edge's metadata, it is easy to check if an edge becomes too out-of-date, allowing it to be removed from the database. This is more straightforward than allowing the LLM to access to the entire database and filtering at the retrieval stage. The downside is that, although some information in an older document may no longer be relevant, such as an old trade idea, other aspects of it may remain useful, such as trends in macro indicators.

In the same fashion as the general-domain documents, the data is split into paragraphs, which make the knowledge graph generation process easier. Since we use research articles as a data source, we encounter charts and tables which can be tedious to describe by hand. Again, GPT-4o may be leveraged to interpret these. Using its multi-modal capability, we can easily transcribe data from a chart or table into text form, and even highlight interesting trends or data points [30].

Lastly, we note that numerical data in text does not integrate well with the knowledge graph structure. There are various methods to overcome this, though we found that our fully fine-tuned embedding model was able to retrieve numerical data well, and this was also reported accurately by the LLM in its final output. However, we consider possible alternatives. One approach is to query a separate live database for specific quantities, like the U.S. 10-Year Treasury Yield, which necessitates function calling or tool interaction. Alternatively, one can prompt an LLM to extract all numerical information from a text and store it in JSON or Python dictionary format, which can then be added back to the LLM's context via a second retrieval step.

2.2.2 Data Sources

We compile only sources which directly relate to or comment on the operations of the U.S. Federal Reserve. This offers a deep insight into this domain. Such documents may include:

- Federal Open Market Committee (FOMC) statements and meeting minutes
- Fed Summary of Economic Projections (SEP)
- Transcripts of speeches by Fed Chair Jerome Powell and other members of the Fed Board of Governors, as well as any following press conferences

- Research articles directly commenting on and forecasting the Fed's actions and U.S. interest rates

By combining both official Fed documents as well as in-house research articles, the domain-specific knowledge graph captures both objective and subjective truths about the topic, which we expect to provide more holistic responses.

2.2.3 Domain-Specific Q&A Pairs

The process of generating Q&A Pairs for the domain-specific documents is largely the same as that for the general-domain documents, although with far fewer documents available. We only consider articles from July and August 2024 to keep them as up-to-date as possible.

2.2.4 Domain-Specific Test Dataset

We have real examples of question and answer pairs in the form of the transcripts from FOMC press conferences. After the speech, reporters are given the chance to ask questions, including follow-ups. This presents an excellent opportunity to scrape genuine Q&A data, without the need for synthetic pairs generated by an LLM. However, since the transcript aims to represent the conversation as accurately as possible, there is often stuttering, extra punctuation and other noise in the text. To remedy this, we again make use of the versatility of LLMs [34] to clean up the text, remove any mentions of names and format it with "Q:" and "A:" denoting each pair. This streamlines the data for use in embedding training. The cleanup prompt is shown in Figure A.5.

Since this is the best source of real-world Q&A data we have, we employ it as our test dataset. We only use press conferences from a maximum of one year ago to keep it relevant, and to avoid the outliers during Covid years. The data is also narrow in scope, relating mostly to questions on monetary policy and the operations and decision-making processes of the Fed. There is, as always, plenty of scope for extension, including using real examples of client interaction with our fine-tuned in-house chatbot, which has access to our entire database of research articles.

Chapter 3

Experiments

3.1 Setup

Before we present our findings, we first elaborate on our methodology for conducting experiments using the PKGE framework. We detail the models used, the datasets and, lastly, the metrics on which we evaluate.

3.1.1 Models

We compare two versions of our procedure against ablated versions, base models and finally against competing solutions.

PKGE Domain-Specific Embedding is the full, final model which has been fine-tuned on current domain-specific data, while PKGE General-Domain is only the first phase of fine-tuning, done on our custom general macroeconomic theory dataset.

The Baseline models are equivalent, except that they are fine-tuned on their respective non-PKGE datasets. The Baseline Simple model ablates our framework entirely, and is fine-tuned on the Baseline general-domain and domain-specific training data at the same time (the Baseline combined dataset). This represents the vanilla way to fine-tune an embedding model for RAG.

The base model we selected for PKGE is multi-qa-mpnet-base-cos-v1, due to its compact size, reasonably large embedding dimension and ease of fine-tuning. We also include multi-qa-MiniLM-L6-cos-v1, a much smaller though less capable embedding model that will illustrate how model size affects performance on our data.

Of course, we include the pre-trained base models to understand the effect of fine-tuning on macroeconomic data for RAG. Lastly, we compare against the state-of-the-art: OpenAI's Text Embedding 3 Small, as well as the finance domain models Voyage Finance 2 and Finance Embeddings Investopedia.

Table 3.1 shows the details of the embedding models we are considering.

All of these have a Sentence Transformer base model, except Text Embedding 3 Small and Voyage Finance 2. The "multi-qa" variants of these models were selected since they have specifically been trained on question and answer pairs and are intended for semantic search applications [36, 35]. Additionally, being a Sentence Transformer, the base model we chose was not too difficult to fine-tune, taking around 2 hours on our own hardware, while still delivering a notable improvement in performance versus the much smaller multi-qa-MiniLM-L6-cos-v1.

3.1.2 Datasets

Choosing suitable datasets on which to evaluate our framework proved to be challenging. The majority of finance-related datasets mostly contain data relating to general and

Model	**Base model**	**Dimension**
PKGE Domain-Specific	PKGE General-Domain	768
PKGE General-Domain	multi-qa-mpnet-base-cos-v1	768
Baseline Domain-Specific	Baseline General-Domain	768
Baseline General-Domain	multi-qa-mpnet-base-cos-v1	768
Baseline Simple	multi-qa-mpnet-base-cos-v1	768
multi-qa-mpnet-base-cos-v1	-	768
multi-qa-MiniLM-L6-cos-v1	-	384
bge-base-en-v1.5	-	768
Text Embedding 3 Small	-	1536
Voyage Finance 2	-	1024
Finance Embeddings Investopedia	bge-base-en-v1.5	768

Table 3.1: The embedding models we use for our experiments.

personal finance topics, equity-related data, such as stock sentiment and form 10-Ks, and general banking questions. These types of tasks are not our goal, but are found in the most common datasets such as FiQA, FinTextQA, and TAT-QA [40, 6, 51]. To preserve some comparability against these, we test on the Finance Alpaca dataset [10], which is a combination of the Stanford Alpaca dataset [39], FiQA dataset [40] and 1,300 pairs generated by GPT-3.5. We consider only the first 10,000 pairs to save on computation time, API calls, and also to avoid the GPT-generated pairs.

We assemble a custom test dataset from Fed press conference transcript PDFs dating back to May 2023, which is our primary focus for testing. The most recent press conference was included in the domain-specific datasets, and previous ones in the Fed press conference dataset for testing. This prevents data leakage. Only press conferences from a maximum of one year ago are considered due to their declining relevance. Covid years are also avoided since they contain non-generalisable data. The Q&A sessions from each press conference transcript are extracted and first pre-processed using OpenAI's GPT-4o [30], as described earlier in Section 2.2.4.

With the Q&A pairs written in a well-formatted text file, we can easily convert this into the required format for our Python libraries to evaluate.

The final list of datasets that we use is shown in Table 3.2. We implement a training-validation split of 80-20, and only train/validate on our own PKGE and Baseline datasets. Fed press conferences and Finance Alpaca are only used for final evaluation and testing.

Dataset	**Purpose**	**No. of Pairs**
PKGE domain-specific	Train/Val	2,360/591
PKGE general-domain	Train/Val	4,000/1,000
Baseline domain-specific	Train/Val	87/24
Baseline general-domain	Train/Val	588/147
Baseline combined	Train/Val	675/171
Fed press conferences	Test	233
Finance Alpaca	Test	10,000

Table 3.2: Our datasets used.

Note again that there is very little available data for the Baseline domain-specific dataset, highlighting the need for the PKGE framework. For the scope of our project, the domain-specific documents include a handful of relevant recent research articles from our database, as well as the most recent Fed summary of economic projections (SEP) and press conference. In production, we would definitely expand the scope of the data to include external research and news articles, as well as capturing a much larger quantity overall. This is discussed further in Chapter 5.

3.1.3 Metrics

We apply the the standard metrics for evaluating retrieval, including accuracy/hit rate, precision, Mean Rank Reciprocal (MRR) and NDCG (Normalised Discounted Cumulative Gain). Accuracy and precision give intuitive understandings for the quality of retrieval, while MRR and NDCG both yield a rank-dependent score. Importantly, in all retrievals, we define exactly one "correct" answer in the corpus (positive pair), with all other answers considered incorrect (negative pairs). This means that the model's recall is identical to its accuracy, so we will only quote accuracy. Also, MRR and NDCG will yield the same interpretation, since there is no possibility of multiple correct retrievals.

For retrieval, we find the top-K closest embedded text chunks in the corpus, in cosine similarity, for $K \in \{1, 3, 5, 10\}$. The corpus contains N queries $\{q_i\}_{i=1,\dots,N}$ and N answers $\{a_i\}_{i=1,\dots,N}$, with $K \leq N$. The positive pairs are the (q_i, a_j) for which $i = j$, and the negative pairs are those with $i \neq j$.

Remark 3.1.1. In practice, there would sometimes be several "correct", or relevant, answers to a given user query, especially when it relates to macroeconomics. This is because there are often many interpretations are possible. It may also be the case that there are more than K relevant answers for any value of K we choose. For simplicity, and given that our aim is to demonstrate the effectiveness of our framework, we consider only one ground truth relevant answer.

Let $\mathcal{R}_{i,K} = \{r_{i,k}\}_{k=1,\dots,K}$ be the retrieved answers for a query q_i, where $r_{i,1}$ is the answer with the highest cosine similarity with the query, and $r_{i,K}$ the lowest. Let $\mathbb{1}_{\{a_i \in \mathcal{R}_{i,K}\}}$ be the indicator function which determines whether or not the positive pair a_i matches any of the retrieved answers. Finally, $\text{rank}(a_i)$ is the position of the positive pair in the ranking of the cosine similarities of the entire answer corpus to query q_i. Of course, $0 < k \leq K$ if a_i is retrieved, and $K < k \leq N$ if not.

We now define the metrics for each K as follows:

$$\text{Accuracy}(K) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}_{\{a_i \in \mathcal{R}_{i,K}\}} \tag{3.1.1}$$

$$\text{Precision}(K) = \frac{1}{N} \sum_{i=1}^{N} \frac{\mathbb{1}_{\{a_i \in \mathcal{R}_{i,K}\}}}{K} \tag{3.1.2}$$

$$\text{MRR}(K) = \frac{1}{N} \sum_{i=1}^{N} \frac{\mathbb{1}_{\{a_i \in \mathcal{R}_{i,K}\}}}{\text{rank}(a_i)} \tag{3.1.3}$$

$$\text{NDCG}(K) = \frac{1}{N} \sum_{i=1}^{N} \frac{\mathbb{1}_{\{a_i \in \mathcal{R}_{i,K}\}}}{\log_2(\text{rank}(a_i) + 1)} \tag{3.1.4}$$

3.2 Fine-Tuning Procedure

PKGE involves a two-phase fine-tuning procedure for embedding models. Firstly, we fine-tune a pre-trained model on general-domain data to produce PKGE-GD, the "generally-trained" embedding. Then, we fine-tune PKGE-GD on domain-specific data to yield PKGE-DS, the final model. The Baseline-GD and Baseline-DS models are trained in the same way, but using the baseline datasets.

3.2.1 Training Objective

The goal of training is to minimise the Multiple Negatives Ranking Loss, defined in 1.1.4. This was also used in the pre-training of our base Sentence Transformer models [36, 35].

3.2.2 Optimiser and Regularisation

The Sentence Transformers library uses the AdamW algorithm as its default optimiser. Based on the Adam optimiser, the AdamW algorithm applies L_2 regularisation to model weights by decoupling it from the gradient descent procedure [22].

The Adam optimiser has become indispensable in the training of machine learning models due to being "straightforward to implement... computationally efficient, [having] little memory requirements... invariant to diagonal rescaling of the gradients, and is well suited for problems that are large in terms of data and/or parameters" [18]. Naturally, then, it is appropriate for our purposes.

Regularisation is a technique which prevents model overfitting by either reducing some weights to zero, or close to zero. We define both L_1 and L_2 regularisation here, adapted from [26].

Consider the Mean Squared Error (MSE) loss function as an example. Let N be the number of observations, Y_i be the actual value, and $\widehat{Y_i}$ be the predicted value.

$$\frac{1}{N}\sum_{i=1}^{N}(Y_i - \widehat{Y_i})^2$$

Definition 3.2.1. (L_1 Regularisation) *An MSE loss function with L_1 regularisation has the form*

$$\frac{1}{N}\sum_{i=1}^{N}(Y_i - \widehat{Y_i})^2 + \lambda\sum_{i=1}^{N}|\theta_i|, \tag{3.2.1}$$

with $\lambda \in [0,1]$ being a chosen parameter and θ_i being the weights.

L_1 Regularisation allows weights to decay to zero, removing its explanatory power, which can promote more efficient, sparser representations of the data using fewer features. This also comes with the disadvantage of potentially underfitting by removing too many features, which can result from a choice of lambda which is too high. L_2 regularisation does not have this particular problem, since it will not fully eliminate features.

Definition 3.2.2. (L_2 Regularisation) *An MSE loss function with L_2 regularisation has the form*

$$\frac{1}{N}\sum_{i=1}^{N}(Y_i - \widehat{Y_i})^2 + \lambda\sum_{i=1}^{N}\theta_i^2, \tag{3.2.2}$$

with $\lambda \in [0,1]$ being a chosen parameter and θ_i being the weights.

The difference with L_2 regularisation is that it does not allow weights to decay to zero, but only to approach zero, leading to non-sparse solutions. L_1 penalises the absolute value of the weights, whereas L_2 penalises their square. In both cases, setting lambda as zero removes the effect of the regularisation component completely.

3.2.3 Training Parameters

All PKGE and Baseline models are trained on the same parameters: 2 epochs of stochastic gradient descent, with a minibatch size of 36 for estimating the gradient. The learning rate is 2e-5, with a warmup given by $0.1 * N *$ epochs, where N is the number of pairs in each dataset.

3.2.4 Hardware

All training and testing is conducted on an Nvidia GeForce RTX 4050, which has 6GB of VRAM. As this is memory-constrained, we choose models which are small enough to fit in the VRAM buffer and would not exceed the capacity during training. This proved capable for our purposes and no cloud compute was necessary, thanks to the relatively low compute requirement of the PKGE framework. For a production-ready embedding model, one should certainly train on more capable hardware, and with significantly more data.

Chapter 4

Results

4.1 Empirical Results

The results on our test datasets are shown in Table 4.1 and Table 4.2.

4.1.1 Fed Press Conferences

Model	**Acc@1**	**Acc@3**	**Acc@5**	**Acc@10**	**Prec@1**	**Prec@3**	**Prec@5**	**Prec@10**	**MRR@10**	**NDCG@10**
mpnet-base	0.3247	0.5022	0.5758	0.6623	0.3247	0.1674	0.1152	0.0662	0.4303	0.4858
MiniLM-L6	0.3247	0.4502	0.5022	0.6104	0.3247	0.1501	0.1004	0.0610	0.4053	0.4535
bge-base	0.2814	0.4589	0.4978	0.5931	0.2814	0.1530	0.0996	0.0593	0.3844	0.4347
Investopedia	0.2641	0.4113	0.4589	0.5455	0.2641	0.1371	0.0918	0.0545	0.3515	0.3976
OpenAI	0.3550	0.4805	0.5541	0.6407	0.3550	0.1602	0.1108	0.0641	0.4393	0.4874
VoyageAI	0.3160	0.5108	0.5671	**0.6753**	0.3160	0.1703	0.1134	**0.0675**	0.4338	0.4918
Baseline-Simple	0.3463	0.5022	0.5541	0.6667	0.3463	0.1674	0.1108	0.0667	0.4391	0.4929
Baseline-GD	0.3463	0.5022	0.5325	0.6364	0.3463	0.1674	0.1065	0.0636	0.4344	0.4823
Baseline-DS	0.3074	0.4589	0.5714	**0.6753**	0.3074	0.1530	0.1143	**0.0675**	0.4112	0.4740
PKGE-GD	0.3593	0.5152	0.5628	0.6494	0.3593	0.1717	0.1126	0.0649	0.4538	0.5006
PKGE-DS	**0.3853**	**0.5195**	**0.5844**	**0.6753**	**0.3853**	**0.1732**	**0.1169**	**0.0675**	**0.4695**	**0.5181**

Table 4.1: Results on the Fed press conferences dataset. The top-performing model is highlighted in bold.

Firstly, PKGE Domain-Specific outperforms all other models on the press conferences dataset, with the exception of ties on Accuracy@10 and Precision@10, which could have been avoided with a larger dataset. On average, PKGE Domain-Specific delivers an 8.92% gain over Baseline Domain-Specific, 6.77% vs Baseline Vanilla and 6.69% vs mpnet-base, its base model, across all metrics. This shows that our framework offers a notable accuracy and ranking improvement.

Phased learning shows mixed results on the dataset, however. Baseline Domain-Specific outperforms on accuracy at high K versus Baseline Simple, but demonstrates worse performance on ranking ability, seen in lower MRR and NDCG scores. This may have been avoided if we had a larger amount of raw domain-specific data available, as it is possible that Baseline-DS was not exposed to enough data.

Notably, Voyage Finance 2 appears to have similar performance to Text Embedding 3, where the former outperforms on $K = 5$ and $K = 10$, while the latter outperforms on $K = 1$ and $K = 3$. Their ranking capability is also very close. Finance Embeddings Investopedia had unexpectedly poor performance, which may be due to insufficient fine-tuning on macroeconomic texts. It also underperforms its own base model, bge-base. Overall, PKGE Domain-Specific delivers the best performance on this dataset, which empirically shows that our model is fit-for-purpose.

4.1.2 Finance Alpaca

Model	**Acc@1**	**Acc@3**	**Acc@5**	**Acc@10**	**Prec@1**	**Prec@3**	**Prec@5**	**Prec@10**	**MRR@10**	**NDCG@10**
mpnet-base	0.2602	0.4487	0.5304	0.6301	0.2602	0.1496	0.1061	0.0630	0.3753	0.4363
MiniLM-L6	0.1947	0.3450	0.4099	0.5028	0.1947	0.1150	0.0820	0.0503	0.2884	0.3395
bge-base	0.2178	0.3821	0.4546	0.5473	0.2178	0.1274	0.0909	0.0547	0.3185	0.3732
Investopedia	0.1674	0.3039	0.3637	0.4507	0.1674	0.1013	0.0727	0.0451	0.2520	0.2993
OpenAI	0.2418	0.4149	0.4927	0.5875	0.2418	0.1383	0.0985	0.0588	0.3477	0.4048
VoyageAI	0.2381	0.4231	0.5044	0.6040	0.2381	0.1410	0.1009	0.0604	0.3505	0.4111
Baseline-Simple	**0.3463**	**0.5022**	**0.5541**	**0.6667**	**0.3463**	**0.1674**	**0.1108**	**0.0667**	**0.4391**	**0.4929**
Baseline-GD	0.2565	0.4413	0.5234	0.6232	0.2565	0.1471	0.1047	0.0623	0.3697	0.4304
Baseline-DS	0.2458	0.4301	0.5089	0.6101	0.2458	0.1434	0.1018	0.0610	0.3575	0.4178
PKGE-GD	0.2407	0.4198	0.4999	0.5977	0.2407	0.1399	0.1000	0.0598	0.3506	0.4097
PKGE-DS	0.2290	0.4018	0.4770	0.5726	0.2290	0.1339	0.0954	0.0573	0.3340	0.3910

Table 4.2: Results on the Finance Alpaca dataset. The top-performing model is highlighted in bold.

On Finance Alpaca, the Baseline Simple model outperforms outright. Firstly, we expect that the increasingly fine-tuned general-domain and domain-specific models should lose some generalisation due to the fine-tuning process. This is especially apparent in the other PKGE and Baseline models, which underperform their own base model, mpnet-base. Finance Alpaca contains a small amount of macro-related data, but mostly consists of personal finance, banking and equities data. Baseline Simple delivers strong outperformance, despite being trained on the same data as the PKGE and other Baseline models. It is possible that, for this dataset, it was more beneficial to expose the model to the full range of data at once, rather than in phases.

Finally, although the OpenAI and VoyageAI embedding models performed in-line with the overall average, they underperform mpnet-base. It may also be possible that some pre-training data for mpnet-base and MiniLM-L6 coincided with this dataset, since both of these had the same training corpus of 215 million Q&A pairs [36, 35].

4.2 Real-World Testing

By far, the most important evaluation for PKGE is testing the entire application on real data, for a real-world task. We build a simple interface which is of the form illustrated in the RAG section of Figure 2.2. We use GPT-4o to generate the final output. Four sample user queries which capture different aspects of the knowledge available in our domain-specific database are included, and the LLM's corresponding responses are detailed in Appendix A.1. Additionally, we detail the LLM system prompt used, which describes how the LLM "consultant" should behave, shown in the appendix in Figure A.6.

The responses demonstrate the efficacy of the PKGE framework. Firstly, PKGE Domain-Specific is easily able to retrieve information on the Fed's potential monetary policy decisions (Figures A.2 and A.3). When asked for potential trade ideas relating to Fed actions, the model is not only able to retrieve the specific text chunk in the database containing the trade details, but also give an explanation to the reasoning for the trade, which is located in a different text chunk (Figure A.1). Finally, our model is still able to deliver a relevant and informative response even when asked an extremely vague question, thanks to its comprehensive system prompt (Figure A.4).

An important consideration for the application is the amount of context to retrieve. Returning only one text chunk is insufficient, while ten is excessive and opens up the LLM to the "lost-in-the-middle" problem. This refers to the phenomenon of worse quality responses when the relevant context is near the middle of a much longer context chunk, with better performance when it is near the beginning or the end [21]. Therefore, we have selected $K = 3$ as our optimal number of texts to retrieve, based on the amount of data we have. With more available data, $K = 5$ may become more appropriate.

Our intended users are researchers or traders at banks, asset managers and hedge funds, who are interested in quickly interfacing with all the current U.S. macro and rates news, as well as our company's views. RAG is especially important for these users because the information must be as true to its source as possible. In production, the system may also be tailored to meet the specific needs of clients, such as information being presented in a certain way. The LLM system prompt may be altered, or additional formatting steps may be introduced to accommodate this, though these extensions are beyond our scope.

Chapter 5

Further Work & Practical Considerations

5.1 Extensions

5.1.1 Generalising PKGE

The PKGE framework is not restricted to macroeconomic data. Although we have shown that it suits this domain well, it could be extended to other popular areas for embedding models and LLMs, such as law, medicine and technical support. In tech support for example, one can train first on general-domain documents such as IT best practices, before then incorporating company-specific procedures, guidelines and tutorials.

Another generalisation of the idea is to extend phased learning, where a model receives the same education progression as humans, beginning with concepts typically taught in primary school and below, and ending with professional-level expertise acquired from decades of experience in a field. This may even be applied at the pre-training stage. We believe both of these could be a compelling area for future research.

5.1.2 Enhancing the RAG Pipeline

We have only considered fine-tuning the embedding model for the retrieval stage of the RAG pipeline, and have not incorporated any fine-tuning of encoder-decoder or decoder-only models. For our purposes, we only require the model to learn efficient representations of our data so that relevant context information can be retrieved reliably, while a pre-trained decoder-only LLM, such as GPT-4o [30], is sufficient in producing the final output to the user.

Alternatively, fine-tuning an encoder-decoder model like Google's T5 [27] may enable better performance overall, since both encoder and decoder weights can be fine-tuned. The PKGE framework eliminates the intensive computation required in fine-tuning an encoder-decoder or decoder-only model, which have significantly more weights. Techniques such as Low Rank Adaptation (LoRA) [13] and Parameter-Efficient Fine-Tuning (PEFT) [23] can alleviate the compute required, however, by only requiring a small number of extra weights to be trained on top of the original model. This yields performance close to that of fine-tuning all model weights, and these techniques may be explored in future work.

5.2 Practical Considerations

In a production setting, our envisaged application should be able to retrieve data and generate responses quickly, as well as being efficient on compute, tokens and API calls. These considerations are central to PKGE, which was built, from the start, to be a commercially viable framework.

Again, we currently focus on the embedding model to improve retrieval and minimise the number of LLM calls needed in the process. General-domain knowledge graph generation is the most token-intensive step, though it only needs to be performed once. This is similar for the domain-specific knowledge graph in terms of Q&A generation to fine-tune the domain-specific embedding model, which only needs to be carried out to keep in-line with new market regimes. LLM calls are required often to update the graph with new edges, though the token usage should be light.

For production, the model must be fine-tuned on considerably more data than we have provided in our demonstration. Our framework can be easily adapted to suit other macroeconomic areas, or even completely different domains. PKGE's knowledge graph Q&A generation allows one to easily generate enough data for decent fine-tuning results, regardless of its area.

The flexibility of PKGE allows it to be implemented into many different kinds of pipelines. Additional steps can easily be added into the process, such as reranking retrieved answers. A reranker model, or cross-encoder, compares the similarity between a query and retrieved document using a full Transformer architecture, which is slow but more accurate than using embedding models [32]. This can overcome the "lost-in-the-middle" problem when retrieving many text chunks. Rerankers are also commonly used in search engines, which could be another area in which PKGE can be adapted.

On balance, although our current PKGE framework has already demonstrated promising performance, there remain many areas which are worth investigating that have the potential to significantly boost its capabilities.

Conclusion

In this paper, we introduced Phased Knowledge Graph-Attuned Embedding: A flexible and commercially applicable framework for enhancing Retrieval Augmented Generation on macroeconomic data.

Factual question answering often requires relevant context to be retrieved to help the LLM to avoid hallucination. Fine-tuning an embedding model for retrieval is done through teaching the model to learn the relationship between queries and their answers [20]. The model then accesses a database to retrieve the most relevant context by finding the top K closest text chunks, in cosine similarity, to an embedded query. In our case, we define K to be 3, which provides sufficient context while minimising the effect of the "lost-in-the-middle" problem [21].

We demonstrated that our fully fine-tuned embedding model, PKGE Domain-Specific, was able to outperform the state-of-the-art Text Embedding 3 from OpenAI, as well as competing finance-specific embedding models, including VoyageAI's Voyage Finance 2, on our custom Fed press conference dataset. This is the closest benchmark to real-world usage of our model. Most importantly, in our test application, we demonstrate high-quality responses to a variety of user queries which all capture the relevant factual information, even when the query is extremely vague.

In addition, PKGE also introduces an effective method of data generation through augmenting existing data via knowledge graphs. Using a pre-trained LLM, one can easily generate new question and answer pairs through grouping existing nodes in unique subgraphs. This can be immensely valuable in domains with very little relevant training data. Interest rates research is one such area; News and opinions quickly become out-of-date as new data arrives. With more information about the domain, the embedding model is able to learn more of its specific language features and topics, leading to increased accuracy at the retrieval stage.

The value in our framework remains in its practical application. To become a valuable resource for traders and researchers in banks, hedge funds and asset managers, the application would require scaling to many concurrent users and implementation on a server with embedding and LLM inference endpoints, along with a live vector/graph database. This paper only proposes the framework, which is now left to the reader to interpret and adapt to their specific use case.

Appendix A

Prompts, Queries & Answers

A.1 PKGE Application Sample Queries & Responses

Query: How do I trade the Fed?
Answer: To trade the Fed based on the view of a single 2024 cut in November or December FOMC meetings, compared to market expectations of more cuts, you can tactically short January Fed Funds. This means you would take a bearish position on January Fed Funds contracts, anticipating a decrease in the Fed Funds rate. The market, however, is pricing in two cuts by September and five cuts by December, showing a more dovish outlook than your single cut expectation.

Figure A.1: Sample LLM response on user query with retrieved context: Fed trade ideas.

Query: How does the Fed see inflation for the remainder of this year?
Answer: The Fed expects inflation to return to its 2 percent goal as the economy progresses. They don't plan to reduce the target range for the federal funds rate until there's more confidence that inflation is moving sustainably towards 2 percent. The Fed will continue to monitor inflation data and make decisions on a meeting-by-meeting basis. They acknowledge the risks of adjusting policy too soon or too late. Despite strong economic fundamentals, growth and inflation have surprised on the downside. The Fed sees balanced risks to employment and inflation and could potentially cut rates once in 2024 to mitigate downside risks to growth, likely in November or December, to prioritize political gains over economic costs.

Figure A.2: Sample LLM response on user query with retrieved context: Fed view on inflation.

Query: Labour market impact on Fed decisions?
Answer: The Federal Reserve closely monitors the labor market conditions when making decisions. Powell stated that they would react quickly to unexpected labor market weaknesses. The recent weak July NFP data suggests labor market rebalancing, but the Fed is likely to frontload its rate cut due to advanced rebalancing. In the labor market, supply and demand conditions have improved, with solid average job gains of 177,000 per month in the second quarter. The unemployment rate is at 4.1%, and overall labor market conditions have returned to pre-pandemic levels. The spread between minority and white unemployment rates signals a strong labor market, with African American/white spread below 2019 average and Hispanic/white spread above and rising due to high migrant inflows.

Figure A.3: Sample LLM response on user query with retrieved context: Labour market impact on Fed decisions.

Query: What's looking interesting now?
Answer: Recent indicators show that economic activity is solid, with strong job gains and a low unemployment rate. Inflation has eased but remains elevated, making modest progress toward the 2% inflation objective. Labor market conditions are improving, with payroll job gains averaging 177,000 per month in the second quarter. The unemployment rate has moved up slightly to 4.1%, reflecting strong job creation in recent years. Wage growth has eased, and the job-to-workers gap has narrowed. Overall, labor market conditions are strong but not overheated. It's important to be cautious with judgments as the pandemic era has been unusual, and traditional indicators may not function as expected.

Figure A.4: Sample LLM response on user query with retrieved context: Vague query.

A.2 PKGE Pipeline Prompts

Clean up this text and remove any conversational utterances, stutters and punctuation. Streamline and simplify the questions but make sure to keep the main ideas, and remove any mention of names. Format the questions and answers like this:
Q: Question text
A: Answer text
The text is given below:
{text}

Figure A.5: Cleanup prompt for pre-processing press conference PDFs.

You are an expert in U.S. rates, U.S. macro and U.S. financial markets. Your role is to write a short analysis piece, in a friendly and conversational tone, by summarising the most important points in the given context.
You prefer to use simple language and very concise but factual answers.
Only use the information provided to answer the query.
Do not use any external knowledge.
Highlight as much factual information from the context as possible, especially numbers, trends and trade recommendations. Present all the facts.
If the context is clearly irrelevant, or the exact information required to answer the question is not available, ignore the context and just tell your client that you are not sure of the answer.
The following are good examples of answers:
Query: What quantities are influenced by the Federal Funds Rate?
Context: Changes in the federal funds rate trigger...(truncated)
Answer: When the Federal Funds Rate changes, it...(truncated)
Query: What do you think about current inflation?
Context: What we're seeing now is better than last year...(truncated)
Answer: I'd like to highlight that the progress on inflation...(truncated)
Also remember that if the user asks you a question not directly related to U.S. rates, macro and financial markets, only respond that your knowledge is specific to those subjects. Otherwise, respond only with your answer.

Figure A.6: LLM system prompt for final output to the user.

Appendix B

Definitions

Definition B.0.1 (Euclidean Norm)**.** [47] Given a vector p, its Euclidean norm is defined as

$$\|p\| = \sqrt{p \cdot p}.$$

Definition B.0.2 (Cosine Similarity)**.** [44] Given two vectors p and q, their cosine similarity is defined as

$$\text{CoSim}(p, q) := \cos(\theta) = \frac{p \cdot q}{\|p\|\|q\|},$$

where θ is the angle between the vectors p and q.

Definition B.0.3 (Euclidean Distance)**.** [46] Given two vectors p and q, the Euclidean distance between them is defined as

$$d(p, q) = \|p - q\|.$$

Bibliography

[1] AI@Meta, *Llama 3 model card*, 2024.

[2] D. Araci, *Finbert: Financial sentiment analysis with pre-trained language models*, 2019.

[3] J. Barnard, *What is embedding?*, December 2023.

[4] Belagatti, Pavan, *A guide to the softmax activation function.* `https://www.singlestore.com/blog/a-guide-to-softmax-activation-function/`, 3 2024. Accessed: 2024-08-31.

[5] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*, O'Reilly Media, Inc., June 2009.

[6] J. Chen, P. Zhou, Y. Hua, Y. Loh, K. Chen, Z. Li, B. Zhu, and J. Liang, *Fintextqa: A dataset for long-form financial question answering*, 2024.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2019.

[8] V. Efimov, *Large language models: Sbert — sentence-bert. learn how siamese bert networks accurately transform sentences into embeddings*, Towards Data Science, (2023). Published on Sep 12, 2023, Accessed: 2024-08-18.

[9] H. Face, *Wordpiece tokenization - hugging face nlp course.* `https://huggingface.co/learn/nlp-course/en/chapter6/6?fw=pt`, 2024. Accessed: 2024-08-07.

[10] Gaurang Bharti, *finance-alpaca (revision 51d16b6)*, 2024.

[11] Google, *Get information about a business or topic with knowledge panels*, 2024. Accessed: 2024-07-12.

[12] M. Honnibal, I. Montani, S. Van Landeghem, and A. Boyd, *spaCy: Industrial-strength Natural Language Processing in Python*, 2020.

[13] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, *Lora: Low-rank adaptation of large language models*, 2021.

[14] IBM, *Named entity recognition*, 2024. Accessed: 2024-07-17.

[15] ——, *What are ai hallucinations?* `https://www.ibm.com/topics/ai-hallucinations`, 2024. Accessed: 2024-08-31.

[16] ——, *What is a knowledge graph?*, 2024. Accessed: 2024-07-12.

[17] R. Jia, M. Lewis, and L. Zettlemoyer, *Question answering infused pre-training of general-purpose contextualized representations*, 2022.

[18] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017.

[19] V. Košár, *Encoder-only, decoder-only vs encoder-decoder transformer architectures*, 2023. Accessed: 2024-08-20.

[20] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. tau Yih, T. Rocktäschel, S. Riedel, and D. Kiela, *Retrieval-augmented generation for knowledge-intensive nlp tasks*, 2021.

[21] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang, *Lost in the middle: How language models use long contexts*, 2023.

[22] I. Loshchilov and F. Hutter, *Decoupled weight decay regularization*, 2019.

[23] S. Mangrulkar, S. Gugger, L. Debut, Y. Belkada, S. Paul, and B. Bossan, *Peft: State-of-the-art parameter-efficient fine-tuning methods.* `https://github.com/huggingface/peft`, 2022.

[24] T. Mikolov, K. Chen, G. Corrado, and J. Dean, *Efficient estimation of word representations in vector space*, 2013.

[25] J. Murel and E. Kavlakoglu, *What is transfer learning?*, Feb 2024.

[26] Neptune.ai, *Fighting overfitting with l1 or l2 regularization*, 2021. Accessed: 2024-08-29.

[27] J. Ni, G. H. Ábrego, N. Constant, J. Ma, K. B. Hall, D. Cer, and Y. Yang, *Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models*, 2021.

[28] R. Nyk, *Graph maker.* `https://github.com/rahulnyk/graph_maker`, 2023. Accessed: 2024-07-10.

[29] OpenAI, *Embeddings - use cases.* `https://platform.openai.com/docs/guides/embeddings/use-cases`, 2024. Accessed: 2024-08-31.

[30] OpenAI, *Hello gpt-4o.* `https://openai.com/index/hello-gpt-4o/`, 2024. Accessed: 2024-07-08.

[31] M. Pfeifer and V. Marohl, *Centralbankroberta: A fine-tuned large language model for central bank communications*, The Journal of Finance and Data Science, 9 (2023), p. 100114.

[32] Pinecone, *A guide to rag rerankers in nlp.* `https://www.pinecone.io/learn/series/rag/rerankers/`, 2024. Accessed: 2024-08-31.

[33] ——, *How to fine-tune sentence transformers with multiple negatives ranking.* `https://www.pinecone.io/learn/series/nlp/fine-tune-sentence-transformers-mnr/`, 2024. Accessed: 2024-08-31.

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

[35] N. Reimers, *multi-qa-minilm-l6-cos-v1.* `https://huggingface.co/sentence-transformers/multi-qa-MiniLM-L6-cos-v1`, 2021.

[36] ——, *multi-qa-mpnet-base-cos-v1.* `https://huggingface.co/sentence-transformers/multi-qa-mpnet-base-cos-v1`, 2021.

[37] N. Reimers and I. Gurevych, *Sentence-bert: Sentence embeddings using siamese bert-networks*, in Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 11 2019.

[38] S. Sudalairaj, A. Bhandwaldar, A. Pareja, K. Xu, D. D. Cox, and A. Srivastava, *Lab: Large-scale alignment for chatbots*, 2024.

[39] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. B. Hashimoto, *Stanford alpaca: An instruction-following llama model.* `https://github.com/tatsu-lab/stanford_alpaca`, 2023.

[40] N. Thakur, N. Reimers, A. Rücklé, A. Srivastava, and I. Gurevych, *Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models*, 2021.

[41] L. van der Maaten and G. Hinton, *Visualizing data using t-sne*, Journal of Machine Learning Research, 9 (2008), pp. 2579–2605.

[42] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, *Attention is all you need*, 2023.

[43] L. Voita, *Seq2seq and Attention — lena-voita.github.io.* `https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html`, 11 2023. [Accessed 21-06-2024].

[44] Wikipedia, *Cosine similarity — Wikipedia, the free encyclopedia.* `http://en.wikipedia.org/w/index.php?title=Cosine%20similarity&oldid=1229946390`, 2024. [Online; accessed 20-June-2024].

[45] ——, *Cross-entropy.* `https://en.wikipedia.org/wiki/Cross-entropy`, 2024. Accessed: 2024-08-31.

[46] ——, *Euclidean distance — Wikipedia, the free encyclopedia.* `http://en.wikipedia.org/w/index.php?title=Euclidean%20distance&oldid=1214273566`, 2024. [Online; accessed 20-June-2024].

[47] ——, *Euclidean space — Wikipedia, the free encyclopedia.* `http://en.wikipedia.org/w/index.php?title=Euclidean%20space&oldid=1221937335`, 2024. [Online; accessed 20-June-2024].

[48] Q. Xie, W. Han, X. Zhang, Y. Lai, M. Peng, A. Lopez-Lira, and J. Huang, *Pixiu: A large language model, instruction data and evaluation benchmark for finance*, 2023.

[49] H. Yang, X.-Y. Liu, and C. D. Wang, *Fingpt: Open-source financial large language models*, FinLLM Symposium at IJCAI 2023, (2023).

[50] Y. Yang, Y. Tang, and K. Y. Tam, *Investlm: A large language model for investment using financial domain instruction tuning*, 2023.

[51] F. Zhu, W. Lei, Y. Huang, C. Wang, S. Zhang, J. Lv, F. Feng, and T.-S. Chua, *Tat-qa: A question answering benchmark on a hybrid of tabular and textual content in finance*, 2021.