LUISS

Corso di laurea in Data Science and Management

Cattedra Machine Learning

Leveraging Vector Embeddings and LLMs for Enhanced Financial Analysis: Dash Application for Central Bank Documents

Prof. Giuseppe F.Italiano

RELATORE

Prof. Antonio Simeone

CORRELATORE

ıone De Benedictis Matr. 764

CANDIDATO

Anno Accademico 2023/2024

Introduction

Natural Language Processing (NLP) is one of the hottest areas of artificial intelligence (AI) thanks to applications like text generators that compose coherent essays, chatbots that fool people into thinking they're sentient, and text-to-image programs that produce photorealistic images of anything you can describe. Recent years have brought a revolution in the ability of computers to understand human languages, programming languages, and even biological and chemical sequences, such as DNA and protein structures, that resemble language. The latest AI models are unlocking these areas to analyze the meanings of input text and generate meaningful, expressive output.

In the era of data-driven decision-making, the ability to process vast amounts of information quickly and accurately has become paramount. Central banks, pivotal in steering national economic policies, are repositories of extensive and complex documents that are critical for policy formulation and economic forecasting. This thesis introduces an innovative approach to harnessing the power of advanced language models through the development of a Dash application designed to interact with a Language Large Model (LLM) fine-tuned with Retrieval-Augmented Generation (RAG) techniques. This application aims to provide an intuitive interface for querying and interacting with specialized documents, specifically those produced by central banks.

The integration of LLMs with retrieval-augmented capabilities represents a significant advance in natural language processing (NLP). Retrieval-augmented generation (RAG) combines the generative capabilities of large language models with the retrieval of document-specific data to generate responses that are not only contextually relevant but also deeply informed by the source material. This hybrid approach enables the model to minimize the risk of hallucinating and to produce outputs that are both precise and contextually enriched, which is particularly useful in handling the complex and nuanced language typical of central bank literature.

This thesis will explore the theoretical underpinnings of LLM and RAG, detailing the mechanisms by which these models integrate and enhance the retrieval of information. Following a comprehensive review of the current landscape of NLP applications in economic contexts, the thesis will present a case study: the development of a Dash application. This application will serve as a practical tool for querying a dataset of central bank documents that have been processed and made accessible through a fine-tuned LLM with RAG capabilities. The project not only highlights the potential of AI in enhancing accessibility to financial information but also provides a scalable model that could be adapted to other domains requiring sophisticated information retrieval systems.

By leveraging cutting-edge AI technologies, this thesis aims to demonstrate how complex information processing tasks can be simplified and made more efficient, thus providing actionable insights more swiftly and accurately. The final objective is to provide a proof of concept that elucidates the benefits and challenges of integrating LLMs with RAG within an interactive application, thereby offering valuable contributions to the fields of artificial intelligence, computational linguistics, and economic policy analysis.

1. Theoretical Foundations

1.1 Natural Language Processing (NLP)

Natural Language Processing (NLP) is the discipline of building machines that can manipulate human language — or data that resembles human language — in the way that it is written, spoken, and organized. It evolved from computational linguistics, which uses computer science to understand the principles of language, but rather than developing theoretical frameworks, NLP is an engineering discipline that seeks to build technology to accomplish useful tasks. NLP can be divided into two overlapping subfields: natural language understanding (NLU), which focuses on semantic analysis or determining the intended meaning of text, and natural language generation (NLG), which focuses on text generation by a machine. NLP is separate from — but often used in conjunction with — speech recognition, which seeks to parse spoken language into words, turning sound into text and vice versa.

The processing of human language is based on understanding the intended meaning of a message, which is difficult even for humans, e.g. when irony is used. All components of natural language, such as **phonetics**, **phonology**, **morphology**, **syntax**, **semantics** and **pragmatics**, must be taken into account in order to gain complete understanding of a message.
Phonetics is about the acoustic properties of a sound produced by the human vocal tract. It examines how sounds are physically constructed, e.g. with the tongue or the lips. The sound of a particular human language is studied by phonology. For example, the English language has 45 distinguishable sounds called phonemes. Phonetics and **Phonology** are particularly important aspects in speech recognition when converting sounds into real words that can be processed by a computer.

Morphology concerns about the meaning and the architecture of words. Stemming and lemmatization, which are described below, are based on this component by transforming words like "going" back to their word stem "go".

The ordering of words and the building of grammatical correct sentences is investigated by the **syntax**.
In contrast, **semantic** examines the meaning of sentences that are constructed using syntax and morphological word forms. To obtain the intended overall meaning of a message, **pragmatics** uses the context of the situation. Therefore, a computer needs to take all parts of natural language into account to use it.

1.2 NLP History

The roots of NLP are often traced back to the **1940s** and **1950s**, beginning with foundational work in linguistics and the hypothesis of Alan Turing, who proposed the renewed Turing Test as a measure of machine intelligence.
This test requires a human to write questions and a computer to generate answers.
For the machine to respond correctly, it must understand the meaning and communicate using natural language. Since then, researchers have investigated several approaches to make computers comprehend and generate texts that seem similar to human language.

In the **1960s**, researchers start to develop NLP systems that relied heavily on hand-coded rules. Linguists would create large sets of rules to describe the structure of languages. At that time, data-driven approaches were not feasible because of the amount of data needed, the high processing overhead and the strong need for efficient learning algorithms. Notable projects from this era include SHRDLU, an early natural language understanding system, and ELIZA, a simulation of a Rogerian psychotherapist, both developed in the 1960s.

In the **1980-1990s**, there was a major shift from rule-based methods to statistical methods. This shift was driven by the availability of larger text corpora and the development of machine learning algorithms. Techniques such as Hidden Markov Models (HMMs) and later Conditional Random Fields (CRFs) were used for tasks like speech recognition and part-of-speech tagging.

In those years the concept of n-grams began to arise. An n-gram is a sequence of n consecutive elements in a text sample, expressed in words or characters. For instance, a tri-gram, consisting of tri words taken from the phrase "machine learning and deep learning", would incorporate "machine learning and", "learning and deep", and so forth. In a similar vein, a character-based tri-gram would be "mac", "ach", "chi", "hin", "ine", "nel" and so on. The idea is to create a series of elements and use statistical techniques to determine the probability of their co-occurrence. This concise yet dynamic definition has spurred the advancement of innovative text generation systems, information retrieval techniques, among others. Applying n-gram definitions has given rise to a fresh approach to document representation that centers on individual words.

Starting in the **2000s,** neural networks begin to be used for language modeling, a task which aims at predicting the next word in a text given the previous words. In 2003, *Bengio et al* [1]. proposed the first **neural language model**, that consists of a one-hidden layer feed-forward neural network. They were also one of the first to introduce what is now referred as word embedding, a real-valued word feature vector in R^d. More precisely, their model took as input vector representations of the *n* previous words, which were looked up in a table learned together with the model. The vectors were fed into a hidden layer, whose output was then provided to a softmax layer that predicted the next word of the sequence. Although classic feed-forward neural networks have been progressively replaced with recurrent neural networks *(Mikolov et al., 2010)* [2] and long short-term memory networks *(Graves, 2013)* [3] for language modeling, they remain in some settings competitive with recurrent architectures, the latter being impacted by "catas- trophic forgetting" *(Daniluk et al., 2017)*[4].

In **2013**, *Mikolov et al.* introduced arguably the most popular **word embedding model**: Word2Vec. Although dense vector representations of words have been used as early as 2003 *(Bengio et al.)*, the main innovation proposed in their paper was an efficient improvement of the training procedure, by removing the hidden layer and approximating the objective. Later that year, they improved the Word2Vec model by employing additional strategies to enhance training speed and accuracy. While these embeddings are not different conceptually than the ones learned with a feed-forward neural network, training on a very large corpus enables them to capture certain relationships between words such as gender, verb tense, and country-capital relations, which initiated a lot of interest in word embeddings as well as in the origin of these linear relationships *(Mimno and Thompson, 2017* [5]; *Arora et al., 2018; Antoniak and Mimno* [6], *2018; Wendlandt et al., 2018* [7]*)*. But what made word embeddings a mainstay in current NLP was the evidence that using pre-trained embeddings as initialization improved performance across a wide range of downstream tasks. Despite many more recent developments, Word2Vec is still a popular choice and widely used today.

In **2015**, *Bahdanau et al.* introduced the principle of **attention**, which is one of the core innovations in neural machine translation (NMT) and the key idea that enabled NMT models to outperform classic sentence-based MT systems. It basically alleviates the main bottleneck of sequence-to-sequence learning, which is its requirement to compress the entire content of the source sequence into a fixed-size vector. Indeed, attention allows the decoder to look back at the source sequence hidden states, that are then combined through a weighted average and provided as additional input to the decoder. Attention is potentially useful for any task that requires making decisions based on certain parts of the input. For now, it has been applied to constituency parsing *(Vinyals et al., 2015)*, [8] reading comprehension *(Hermann et al., 2015*) [9], and one-shot learning *(Vinyals et al., 2016)*[10]. More recently, a new form of attention has appeared, called self-attention, being at the core of the Transformer architecture. In short, it is used to look at the surrounding words in a sentence or paragraph to obtain more contextually sensitive word representations.

The real revolution began in **2017** with the advent of the transformer model, introduced in the seminal paper "Attention is All You Need" by Vaswani et al. in 2017.[11]

Unlike RNNs and LSTMs, which processed data sequentially, transformers used a mechanism called ***self-attention***. This allowed the model to weigh the importance of different words in a sentence, regardless of their position, enabling it to capture long-range dependencies more effectively.
The transformer architecture consists of an encoder and a decoder, similar to the seq2seq model, but with a crucial difference: the reliance on self-attention mechanisms. This design choice not only improved the model's ability to handle long-range dependencies but also significantly increased its parallelization capabilities. As a result, transformers could be trained on larger datasets and at a faster pace than their predecessors.

The success of transformers over LSTM and seq2seq models can be attributed to several factors:

1. **Handling Long-Range Dependencies:** Transformers can capture relationships between words in a sentence, regardless of their distance, more effectively than RNNs and LSTMs.
2. **Parallelization:** The self-attention mechanism allows for parallel processing of input sequences, leading to faster training times and the ability to handle larger datasets.
3. **Scalability:** Transformers are highly scalable, enabling the development of large-scale pre-trained models that can be fine-tuned for various tasks.
4. **Flexibility:** The transformer architecture can be adapted for a wide range of NLP tasks, from text generation to language understanding.

In conclusion, transformers have fundamentally changed the landscape of natural language processing. Their ability to efficiently process sequential data, handle long-range dependencies, and scale to large datasets has established them as the go-to architecture for NLP tasks. As research continues, we can expect further innovations and refinements in transformer models, solidifying their position as a cornerstone of modern NLP.

1.3 Feature selection and Preprocessing

Feature selection and preprocessing are significant tasks in Artificial Intelligence, especially in NLP, this task does have tremendous impact on the success of text analysis. This is mostly caused by the unstructured and arbitrary nature of text data. Furthermore, machines need structure and numerical data. A couple of approaches for this transformation task, e.g. word embeddings or the vector space model, exist. This section's scope lies on the theoretical foundation of different preprocessing and feature selection techniques.

1.3.1 Tokenization

For processing written natural language it is inevitable to split texts into smaller units, which are called tokens. Computers need to distinguish single entities of a text and tokenization is used to create them. Usually tokens represent simple words, which are the smallest independent units of natural language, but tokens could be also subwords, entire sentences or characters.

Tokenization is crucial for several reasons:

- **Simplifies Text**: It breaks down large texts into manageable units, making further processing like parsing and syntactic analysis easier.
- **Enables Vocabulary Creation**: Most NLP models work with numerical data, so tokens are often converted into numerical representations. Tokenization is the first step in creating a vocabulary of unique tokens that can be mapped to numbers.
- **Facilitates Feature Extraction**: Many NLP features, such as n-grams, depend on tokens. Accurate tokenization allows for effective extraction of these features, which are critical for tasks like machine translation and text classification.

By using tokens, so-called n-grams can be created, which indicate a token set with the length of n. "Gramma" is the Greek word for letter or token. When talking about a set of n letters in words, it is about character n grams.

1.3.2 Stop Word Removal

Stop word removal is a preprocessing technique used in natural language processing (NLP) that involves eliminating words from text data that are deemed to be of little value in contributing to the understanding of the content. These words, commonly referred to as "stop words," typically include articles, prepositions, conjunctions, and some common verbs and adjectives. The rationale for removing stop words stems from the observation that they occur frequently in the language but generally do not carry significant semantic weight or contribute to the distinctiveness of the text in analytical tasks such as topic modeling, sentiment analysis, and classification.

Furthermore, stop word removal can enhance the efficiency of NLP systems by decreasing the amount of data that needs to be processed, thereby speeding up computations and reducing resource consumption.

However, it is important to approach the elimination of stop words with caution, as overzealous removal can lead to the loss of important information. For instance, the removal of negations such as "not" can change the sentiment of a sentence, thereby affecting the accuracy of sentiment analysis models.
The selection of stop words can vary depending on the language and the specific requirements of the task. Custom stop word lists may be developed to better align with the linguistic characteristics of the domain-specific texts being analyzed. In academic settings, this practice is underscored by the need to adapt preprocessing techniques to the unique challenges and objectives of scholarly research, ensuring that the integrity and subtlety of the original texts are maintained while optimizing the analytical process.

1.3.3 Stemming

Besides stop word elimination, stemming is a useful technique to map words to their word stems and further reduce the input dimension. This helps to extract the real meaning of a text and makes the unstructured data better accessible for a machine.
The process, known as "stemming," involves the algorithmic truncation of derivatives of a word to a common stem. This enables various forms of a word, such as "connect," "connected," "connecting," and "connection," to be analyzed as their root form, "connect."
Stemming algorithms typically employ simple heuristic processes that strip suffixes from words based on common morphological and inflectional endings in a language. For example, stemming might remove suffixes like "-ing," "-ly," "-es," "-s," and others to focus on the root word. This method is particularly useful in search contexts where the intent is to match on the core meaning of words rather than their specific grammatical uses. It simplifies and speeds up retrieval processes by reducing the total number of distinct terms that must be indexed.
Despite its utility, stemming is often criticized for its lack of linguistic sophistication. Because it relies on crude heuristic rules, stemming can sometimes lead to errors known as over stemming and under stemming. Over stemming occurs when words are reduced too much, leading to the loss of meaningful distinctions between terms (e.g., "universe" and "university" both stemmed to "universe"). Under stemming happens when words that should be reduced to the same root are not, due to the limitations of the heuristic rules. These errors can reduce the precision and recall of retrieval systems.

1.3.4 Lemmatization

Lemmatization in natural language processing (NLP) is the process of reducing words to their base or dictionary form, known as the lemma. This technique considers the morphological analysis of words, making it more precise than simple stemming. Lemmatization involves understanding the word's part of speech and its context to correctly identify its canonical form. This method is essential for tasks where accurate and meaningful text analysis is required, such as in information retrieval and academic research, as it ensures that different forms of a word are analyzed as a single entity. Overall, lemmatization enhances the accuracy and effectiveness of text processing applications by maintaining semantic consistency.

1.3.5 Vector Space Model

Besides, preprocessing the words themselves, their representations must be changed into a machine-readable format. The Vector Space Model is an approach that transforms a text into one vector. It is based on one-hot-encoding of words. Given a set of textual documents (corpus), it is possible to create a vocabulary with the length of N. The one-hot-encoded word vector represents a word by 1 at the corresponding vocabulary entry. For example, if the term “apple” is the i-the unique word in the corpus, the vector has the length of N and a 1 in the i-th position, all other entries are 0:

$$one - hot(apple) = (0, \dots ,1, \dots ,0) \quad \in N^N$$

The vector space model extends this model to documents. Any document d can be mapped to its vector space representation using the function Ψ. The vocabulary's distinct terms, also known as terms, are represented by the characters t1,..., tn.

$$\psi\colon d \mapsto \vec{\psi}(d) = \left(\mathrm{tf}(t_1, d), \mathrm{tf}(t_2, d), \dots, \mathrm{tf}(t_n, d)\right) \in R^N$$

Ψ counts the occurrence of each term (tf) in the vocabulary per document. Therefore, the document vector can have more than one entry that is not 0. The function $tf(t_1, d)$ specifies how often the i-th vocabulary word appears in the document d.

By using the Vector Space Model, the sparsity of the document vectors could be one major problem since N − length(d) positions are 0. Another issue is that the distance between two different document vectors is very small (curse of dimensionality). Therefore, it is not easy to distinguish between documents, especially when it comes to grouping similar documents, e.g. in clustering. Besides simply using the term frequency (tf), it is also possible to give every word a weighting, according to its relative appearance in the corpus. This could be done by using term frequency divided by Inverse Document Frequency (tf-idf), which gives less meaning to common words in a corpus.

However, one inherent challenge of VSM is the propensity for document vectors to exhibit sparsity, as the dimensionality *N* of the space—equivalent to the number of unique terms across all documents—is typically vast, and any single document only contains a small subset of these terms. This sparsity is compounded by the curse of dimensionality, which can render distance measures between high-dimensional vectors less meaningful and can particularly obfuscate the process of clustering similar documents.

Principal Component Analysis (PCA) or Latent Semantic Analysis (LSA) offer a solution to this conundrum by enabling dimensionality reduction. For instance, through the identification and transformation onto the directions that best represent the variance in the data, PCA retains the essence of the dataset with fewer dimensions. This transformation is achieved by eigen decomposition of the data's covariance matrix or singular value decomposition of the data matrix itself. The resultant dimensions, or principal components, are linear combinations of the original variables and are orthogonal to one another, ensuring that they represent independent sources of variance in descending order of significance.
The application of PCA to a term-document matrix effectively reduces the number of dimensions, thereby mitigating the issue of sparsity and facilitating computational efficiency. Moreover, in the reduced dimensional space, similarities and differences between documents can become more pronounced, aiding in tasks such as clustering.

However, while PCA reduces dimensionality and can somewhat preserve the overall variance of the data, it is not without its drawbacks. The technique relies on linear transformations, which may not always capture the nonlinear relationships between terms in natural language. Additionally, the 'principal components'—being linear combinations of possibly thousands of original terms—can be challenging to interpret in a meaningful way with respect to the original textual content.

Furthermore, the embeddings created by PCA, while compact, may not adequately capture the semantic richness of the documents. They provide a view that prioritizes variance over the preservation of local structures and contextual nuances that are often crucial in NLP tasks. Consequently, while PCA can streamline data and expose broader patterns, the resultant embeddings may not serve well for tasks that rely on the subtle and context-dependent interplay of terms within the text.

1.4 Deep NLP

With the advent of deep learning, natural language understanding techniques and methodologies have undergone significant transformations. Deep learning models have demonstrated remarkable effectiveness in various NLP tasks, particularly in semantic understanding. Unlike conventional methods, these models prioritize the encoding of textual information into latent vector spaces that capture contextual subtleties.

1.4.1 Semantic Embeddings

Aware of the prior problems of text vectorization techniques, deep learning's generalization power is utilized in the development of the next generation of text embedding. It relies on the distributional hypothesis, which assumes that words with similar meanings will frequently occur in similar textual contexts.

The first attempt to apply this hypothesis to NLP was LSA, but further research was required to determine the semantic relationships between words.
Words with similar meanings can now be mapped to comparable vectors thanks to machine learning models that can represent words as vectors in high-dimensional spaces thanks to advances in computing power and theoretical frameworks.

One of the most known models to compute semantic embeddings is *Word2Vec,* this model was introduced in 2013 by *Mikolov et al. in 2013*.
Word2Vec utilizes a neural network model targeting to capture words' meanings and contexts in a vector space of high dimensions. The primary assumption is that each word relies on the surrounding words; this denotes the C context, which includes the k words encompassing the target word w.
Embeddings of word2vec do not have the shortcoming of one-hot encoded word vectors, which can only recognize whether a word is exactly the same or not. Word2Vec makes these differentiations more distinctive by including a word's antecedents and successors.

The word representations include semantic and syntactic information, which is retrieved from the context words. The figure below illustrates the position of word vectors with three dimensions. The arrows indicate a mathematical distance between two words. For instance, the cosine distance is a suitable function for calculating the difference between word vectors. In an embedding space, it is possible that same relations between words can be represented by a similar distance and direction. For example, the words "Man" and "Woman" do have a similar distance to "King" and "Queen". This means that the contextual difference between "Man" and "Woman" is comparable to "King" and "Queen". Words that correspond to the male gender are used in the context of "Man" and "King", whereas "Woman" and "Queen" are surrounded by more female word forms. Thus, the syntax and semantic of a word is incorporated into a word representation. This is the real strength of word embeddings or word2vec. It makes it easier to find synonyms and extract a speaker's intended meaning more easily.

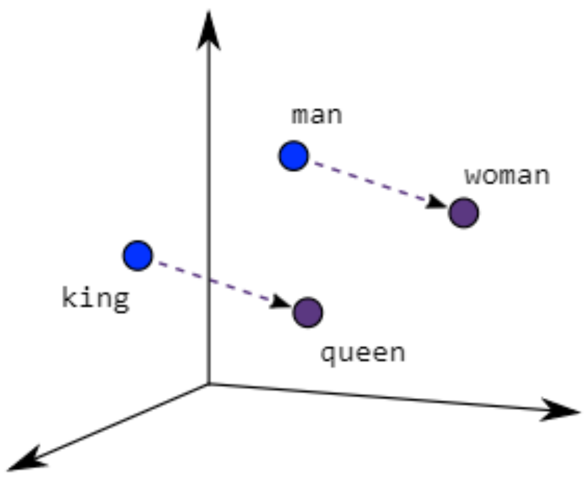
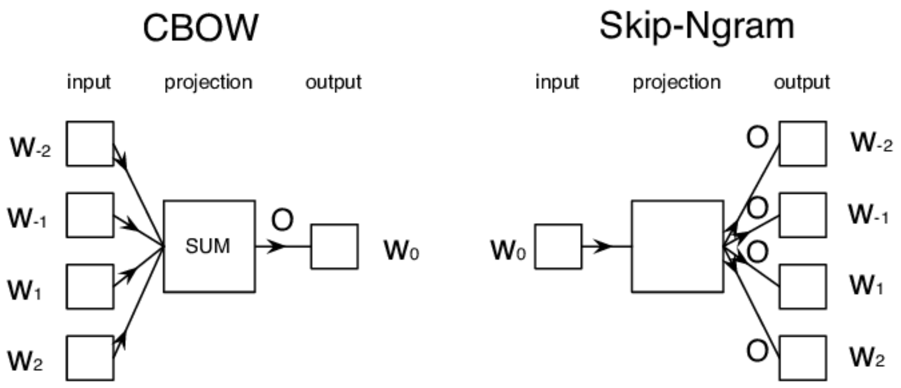


Figure 1. Word2Vec Semantic Embeddings

Furthermore, Word2Vec may exploit either of two model architectures:

1. **Continuous Bag of Words (CBOW)**: The CBOW model predicts the current word based on the context. During training, the model looks at a window of surrounding words (the context) and tries to predict the word in the middle of the window. The order of words in the context does not influence prediction (hence the term 'bag of words').
2. **Skip-Gram**: The Skip-Gram model works in the reverse manner of CBOW. It uses the current word to predict the surrounding context. Given a word in the middle of a window, it predicts the likelihood of each word in the vocabulary appearing in the surrounding window.



.

The model consists of a single hidden layer that produces the vector representation of words; these vectors are initiated with random values and gradually updated during training. These word embeddings contain important semantic information that enables the detection of relationships between words. To understand how similar are two words (w_i, w_j) the similarity between their vector representations ($\vec{w}_i$, $\vec{w}_j$) can be measure. Different measures can be used like Euclidean, Manhattan or cosine similarity to accomplish this task. Each has its advantages and disadvantages regarding semantic search.

The cosine similarity of two vectors calculates the cosine of the angle between:

$$\cos(\vec{u_i}, \vec{u_j}) = \frac{\vec{u_i} \cdot \vec{u_j}}{|\vec{u_i}||\vec{u_j}|}$$

This method is popular because it can easily be used to determine whether two words are similar (cosines close to 1), unrelated (cosines close to 0), or opposite (cosines close to -1). Also, it stays constant regardless of the length of the two vectors, it is computationally efficient for scattered vectors.

1.4.2 Neural Network Architectures: Feed-Forward and RNN

The first neural networks used for NLP tasks were Recurrent Neural Networks (RNN), RNNs introduced the capacity to handle sequences of data, processing input data in a serial manner and maintaining a form of memory through their hidden states.
They differ from traditional neural networks as the next element is assumed to depend on all the previous elements in the sequence. This sequential nature makes them ideal for text processing since they can consider previous words when generating the next.

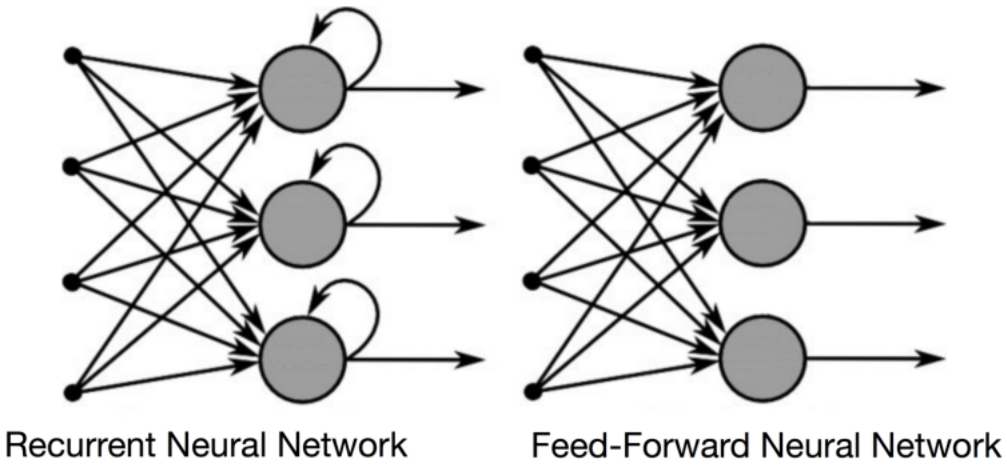


Figure 2 Feed-forward and recurrent architecture comparison

Although this architecture can process long sequences of text, one problem of standard RNNs is the problem of vanishing and exploding gradients. The first problem makes it difficult for RNNs to keep track of the dependencies in longer input sequences.
This is due to two reasons. Firstly, the hyperbolic tangent and the sigmoid function, which are often used in RNNs for activation, saturate very quickly and so their gradient gets closer to 0. Secondly, by applying BPTT the gradient is exponentially reduced by multiplying it with the recurring weight matrices. This also causes the gradient to converge to zero very fast. The phenomenon of the exploding gradient leads to high oscillation of the network’s weights and increases learning time, which could lead to network failure.

1.4.3 Long Short-Term Memory Network (LSTM)

To address the shortcomings of RNNs, Long Short-Term Memory networks (LSTs) were developed. LSTMs incorporated a more complex structure with a series of gates that regulated the flow of information. These gates controlled how information was updated, forgotten, and outputted at each step of the sequence, allowing LSTMs to retain information over longer sequences than typical RNNs. This architecture significantly mitigated the vanishing gradient problem, making LSTMs much more effective for a wide range of tasks, including machine translation, speech recognition, and text summarization.

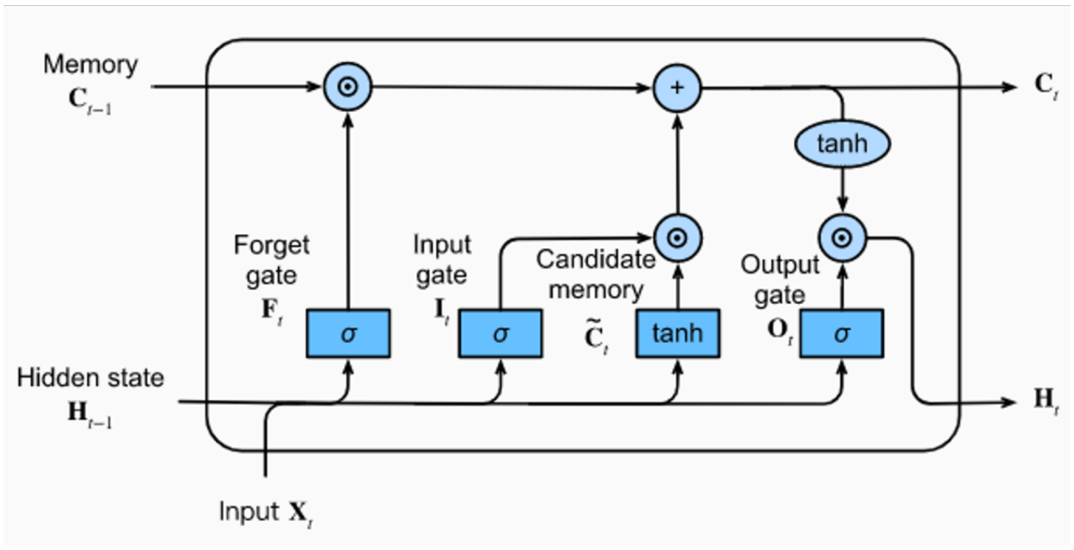

Memory $\mathbf{C}_{t-1}$
$\mathbf{C}_t$
tanh
Forget gate $\mathbf{F}_t$
Input gate $\mathbf{I}_t$
Candidate memory $\tilde{\mathbf{C}}_t$
Output gate $\mathbf{O}_t$
σ
σ
tanh
σ
Hidden state $\mathbf{H}_{t-1}$
$\mathbf{H}_t$
Input $\mathbf{X}_t$

Figure 3 LSTM Architectural structure

The key to LSTMs' effectiveness is their internal structure, which includes several gates that control the flow of information. Each LSTM unit consists of a cell state and three gates:

1. **Forget Gate**: This gate decides what information is discarded from the cell state. It looks at the previous hidden state and the current input, passes it through a sigmoid function, and outputs a number between 0 and 1 for each number in the cell state, where 1 indicates "completely keep this" while 0 indicates "completely get rid of this."
2. **Input Gate**: The input gate updates the cell state with new information. It first decides which values to update using a sigmoid function, and then creates a vector of new candidate values that could be added to the state. A pointwise multiplication between the sigmoid gate output and the candidate values allows for the update.
3. **Output Gate**: The output gate decides what the next hidden state should be. The hidden state contains information on previous inputs. The hidden state is also used for predictions. The sigmoid function decides which parts of the cell state make it to the output, and then a tanh function gives a weightage to the cell state, making it between -1 and 1, and multiplies it by the output of the sigmoid gate, so that we only output the parts we decided to.

Despite their effectiveness, LSTMs are not without limitations. They can be computationally intensive and prone to overfitting, especially on smaller datasets. They also have difficulty with very long dependencies and can be outperformed by newer architectures like the Transformer on some tasks. Nonetheless, for a significant period, LSTMs represented the state-of-the-art in sequence modeling and are still used in many applications today.

1.4.4 Transformer

As we said in the NLP History paragraph, until 2017, the best solution for encoding text sequences were RRN or LSTM.
The shift from LSTM-based architectures to Transformer models in natural language processing represents a significant paradigm change. This shift is grounded in the Transformer's ability to handle sequential data differently and more effectively.
Transformers, introduced by Vaswani et al. in the paper "Attention is All You Need" in 2017, moved away from the sequential data processing used by RNNs and LSTMs. Instead of analyzing data in order, Transformers process entire sequences simultaneously. This is made possible by the self-attention mechanism, which allows the model to weigh the importance of different parts of a sequence in relation to each other, no matter how far apart they are in the sequence.

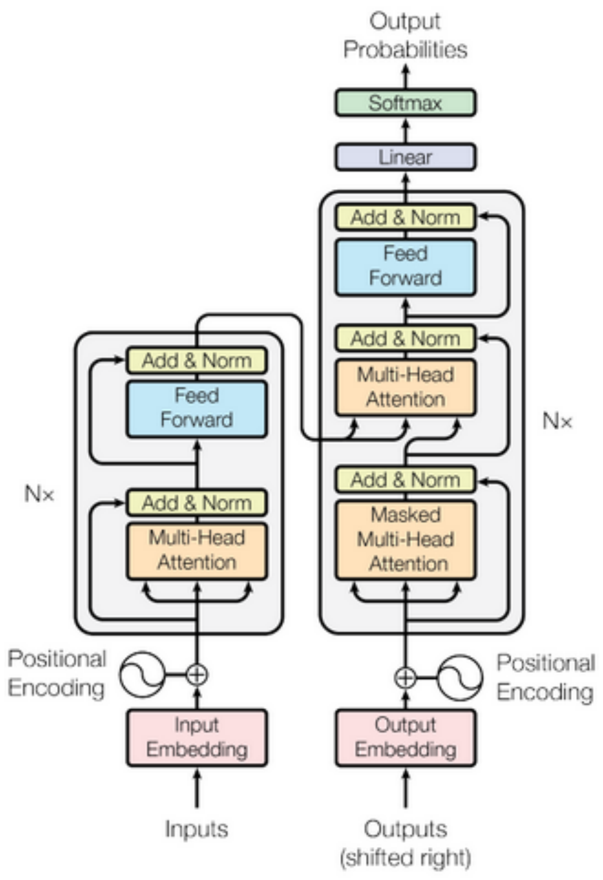


Figure 4 Transformer Architecture

The Transformer is a sequence-to-sequence (seq2seq) model that takes a sequence of items (words, letters, …) from one domain and converts it into a sequence from another. For example, we input a sentence to obtain another, as in translating from one language to another. In Fig. 4, we observe that the transformer comprises an encoder-decoder containing two blocks. On the left, there is the encoder block; on the right, there is the decoder block.
The advantages of Transformer are:

1) **Parallelization**: Unlike RNNs and LSTMs, which require sequential processing, Transformers allow for much greater parallelization during training because they do not depend on the computations of the previous step. This characteristic significantly speeds up training and makes it feasible to scale up models to process vast quantities of data.
2) **Long-Range Dependencies**: Attention mechanism enables the Transformer to grasp the dependencies between the sequence elements, weighing them in such a way as to give each of them a different importance. To do so, it uses learnable weights during the training phase.
   Let us consider a sentence consisting of n words S: w1, w2, . . . wn, where each element w_i is represented through an initial x_i and a triplet of vectors q_i, k_i and v_i, respectively: "query", "key", and "value" vectors that are used when calculating the attention of that specific word. The triplet of vectors is obtained by multiplying the initial matrix of weights of W^q , W^k and W^v, with the vector x_i . To score each word that is part of the same sequence against the word itself w_i (from here the name of self-attention), the dot product is performed between qi and ki followed by a softmax operation. Finally, the scores are multiplied by the vector of values v_i

   To further enhance the self-attention layer, the multi-headed attention mechanism is employed. This consists of n weight matrix W^q , W^k and W^v, all randomly initialised. Everyone will acquire distinct relationships, resulting in varied representations combined through another layer of trainable weights W^0.

3) **Versatility and Scalability**: Transformers are highly versatile and have been adapted for a wide range of NLP tasks, such as language understanding, translation, summarization, and generation. They have shown remarkable performance when scaled, with larger models generally achieving better performance on various benchmarks.
4) **Transfer Learning**: Transformer models, especially large pre-trained models like BERT, GPT, and their successors, have enabled transfer learning in NLP, where a model is pre-trained on a large corpus of text and then fine-tuned on smaller, task-specific datasets. This has led to state-of-the-art performances on a wide array of NLP tasks.

Encoder

The encoder in a Transformer model is responsible for parsing the input and crafting a comprehensive representation that the decoder uses to create the output. Structurally, the encoder is composed of several identical layers, each with two main components.
First, there's the multi-head self-attention mechanism, which allows the encoder to examine various parts of the input in parallel and assign significance to words based on their contextual relevance.

Then, we have point-wise feed-forward networks. The data from the self-attention mechanism is fed into this network, which includes two linear layers with a ReLU activation in the middle. This setup processes the input data through learnable parameters.
Furthermore, the architecture incorporates residual connections, which combat the vanishing gradient problem, facilitating effective training. Normalization layers are also included to ensure the outputs are normalized, maintaining a consistent mean and variance.
As a result, for every word in the input, the encoder outputs a d-dimensional vector that encapsulates its contextual information.

Decoder

Regarding the decoder block, its purpose is to generate the final output. It closely resembles the encoder but includes an additional component: the multi-head cross-attention mechanism. This part focuses on the encoder's output, using the decoder's query matrix alongside the encoder's key and value matrices in its computations. A crucial difference in the decoder is the masked multi-head self-attention, which ensures that each position can only attend to preceding positions, preserving the model's auto-regressive characteristic, ensuring that the prediction only depends on those output tokens that have been generated.

1.4.5 Attention

In large language models (LLMs), the attention mechanism serves as a computational framework that enables the model to dynamically allocate computational resources towards the most relevant parts of the input data. This mechanism is crucial for managing long-range dependencies within the input sequences, which is especially beneficial for tasks that involve understanding the context or generating coherent language outputs.

The technical operation of attention mechanisms in LLMs can be explained as follows: Attention functions compute a weighted sum of values, where the weight assigned to each value is computed by a compatibility function of the corresponding key with the query. This process is described mathematically as:

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

Here, Q, K, and V represent the queries, keys, and values respectively—these are vectors obtained from the input data through linear transformation. d_k is the dimensionality of the keys, and the scaling factor $\sqrt{d_k}$ is used to avoid overly large values of the dot products that could push the softmax function into regions where it has extremely steep gradients, leading to instability in learning. The softmax operation ensures that the weights sum to one, effectively allowing the model to focus by assigning higher weights to more relevant data.
The flexibility of the attention mechanism is further exemplified in its different forms, notably the "self-attention" used within the Transformer architecture. In self-attention, all keys, values, and queries come from the same previous layer, which allows the model to integrate information across the entire input sequence. This architecture's effectiveness is enhanced through multi-headed attention, which involves parallel attention layers with different, learnable linear transformations for queries, keys, and values. This setup allows the model to capture various aspects of the input data in different representational subspaces, thereby enriching the model's ability to discern fine-grained dependencies and semantic nuances (Galassi, Lippi, & Torroni, 2019; DeRose, Wang, & Berger, 2020).

2. Natural Language Generation (NLG) and LLM

Natural Language Generation (NLG) is a subfield of artificial intelligence and computational linguistics that focuses on the development of systems capable of generating coherent and contextually relevant text in human languages from structured data or conceptual information. The primary aim of NLG is to facilitate seamless human-computer interaction by allowing machines to produce textual content that is both understandable and appropriate for the given context, essentially enabling computers to communicate ideas and information as a human would.

The process of NLG can be broadly decomposed into several core stages, which typically include content determination, discourse planning, sentence planning, and finally, text realization. Each of these stages plays a crucial role in ensuring that the generated text meets the specific communicative goals set for the system.

1. **Content Determination**: This stage involves selecting the relevant information to be included in the generated text. It requires the system to assess the importance and relevance of different pieces of information in relation to the communicative goals and the needs of the end-user.
2. **Discourse Planning**: Also known as document structuring, this phase involves organizing the selected content into a coherent structure. It determines the logical flow of information across multiple sentences or paragraphs, ensuring that the text is logically sequenced and easy to follow.
3. **Sentence Planning**: This stage includes the tasks of lexicalization (choosing the specific words to use), referring expression generation (deciding how to refer to entities), and aggregation (combining information into single sentences where appropriate). The goal here is to frame the structured information into linguistically correct and coherent sentences.
4. **Text Realization**: The final stage involves converting the planned sentences into fluent natural language text. This typically involves grammatical adjustments and refinements to enhance the readability and naturalness of the output.

The challenges in NLG revolve around the generation of text that is not only grammatically correct and semantically coherent but also contextually appropriate and engaging for the user. Recent advancements in deep learning, particularly with models like the Transformer and techniques such as sequence-to-sequence learning, have significantly improved the quality of text generation, enabling more dynamic and context-aware outputs.

2.1 GPT

The Generative Pre-Trained Transformer (GPT) is a state-of-the-art deep learning model designed to perform a variety of natural language processing tasks. Developed by OpenAI, GPT is grounded in the Transformer architecture, which relies heavily on self-attention mechanisms to process sequences of data. This architecture has revolutionized the field by enabling models to consider the entire context of a sentence or a paragraph, which is crucial for generating coherent and contextually relevant text.

Technically, GPT is distinguished by its pre-training and fine-tuning stages. During pre-training, the model is exposed to a large corpus of text and learns to predict the next word in a sentence, given the words that preceded it. This unsupervised learning phase allows GPT to develop a broad understanding of language, grammar, and world knowledge.

In the fine-tuning stage, GPT is then adapted to specific tasks such as text summarization, question answering, or any task requiring natural language understanding or generation. This is achieved by continuing the training process on a smaller, task-specific dataset, allowing the model to refine its capabilities according to the specific requirements of the task.

The model's effectiveness stems from the Transformer's ability to handle long-range dependencies in text, meaning it can maintain coherence over longer passages than prior models like RNNs or LSTMs. This capability is enhanced by the attention mechanism, which dynamically adjusts to focus more on relevant parts of the input data as needed.

GPT's architecture features several layers of Transformer blocks, where each block contains a multi-head self-attention layer followed by a position-wise fully connected feed-forward network. Normalization and residual connections are also employed within each block to aid in training deep networks.

The scalability of GPT is evident from its various versions, In its first version, GPT-1 was constructed with a stack of 12 Transformer decoder blocks, generating 117 million parameters. Pre-training was conducted on a dataset named BookCorpus[39], which comprises roughly 4.5GB of text derived from 7000 books. These two features give the model a comprehensive and abundant linguistic foundation capable of comprehending intricate language structures. The latest version of GPT-3 reaches a size of 175 billion parameters (requiring 800 GB of storage), allowing the model to produce sophisticated text almost indistinguishable from human-generated text.

2.1.1 GPT 3.5

The Generative Pre-Trained Transformer 3.5 (GPT-3.5) is an advanced iteration of the GPT-3 model, maintaining the foundational Transformer architecture and training datasets, with a critical enhancement through the integration of Reinforcement Learning with Human Feedback (RLHF). This enhancement refines the model's outputs and operational ethics significantly.

RLHF is a sophisticated training approach that leverages direct human interactions to refine the learning algorithms of machine models. In this methodology, human feedback serves as a dynamic guide to reinforce desirable outputs and mitigate undesirable ones. Feedback can be either positive, to reinforce correct model behaviors, or negative, to correct errors or undesirable responses. This direct intervention allows for precise adjustments to the model's parameters and aids in labeling previously unannotated data effectively.

Human trainers play a pivotal role in this process; they evaluate the responses generated by the model, categorizing them as appropriate or inappropriate. This feedback loop is instrumental in reducing the generation of biased, toxic, or irrelevant content by the model, thereby enhancing its reliability and safety. Specifically, RLHF facilitates the elimination of outputs that could be harmful or unsafe, such as responses that involve illegal activities, offensive content, or personal data misuse.

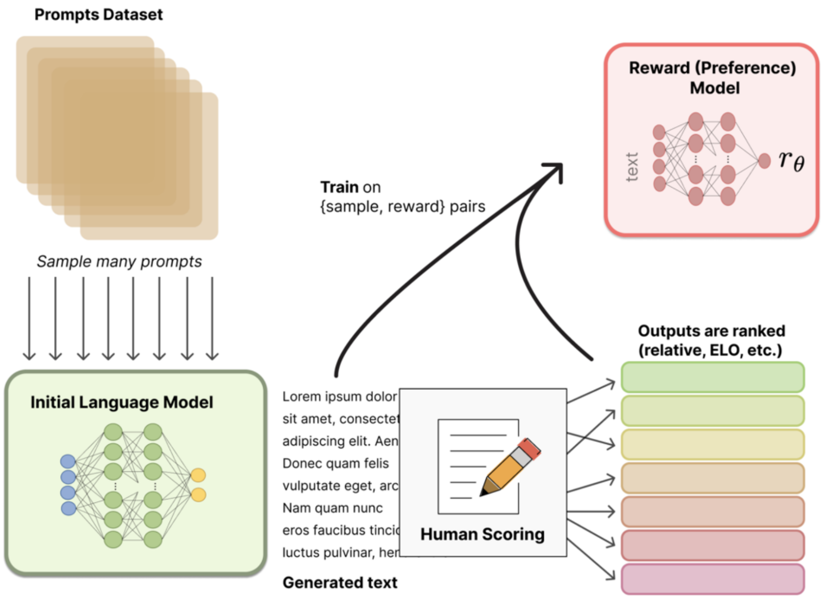

Prompts Dataset
Sample many prompts
Initial Language Model
Train on {sample, reward} pairs
Reward (Preference) Model
text
r_θ
Outputs are ranked (relative, ELO, etc.)
Lorem ipsum dolor sit amet, consectet adipiscing elit. Aen Donec quam felis vulputate eget, arc Nam quam nunc eros faucibus tincid luctus pulvinar, her
Human Scoring
Generated text

Figure 5 Reinforcement Learning from Human Feedback process

A significant focus of the GPT-3.5 development was optimizing the model for enhanced dialogue capabilities. This focus is evident in the deployment of ChatGPT, a conversational agent that utilizes GPT-3.5 to interact with users. ChatGPT is designed to handle a broad spectrum of conversational tasks, providing responses within the scope of its extensive training data. This data encompasses a comprehensive 570GB corpus, including diverse sources like Web Text, Common Crawl data, Wikipedia entries, and an assortment of books.

The RLHF methodology not only improves the model's ability to generate contextually appropriate and safe dialogue but also enhances its functionality across various user interactions. By incorporating human judgments directly into the training loop, GPT-3.5 is better equipped to understand and adapt to the nuances of human language and the subtleties of conversational context.

The incorporation of RLHF in GPT-3.5 training represents a forward leap in developing AI systems that are both ethically aware and contextually adaptive. This training approach offers a robust framework for future models, potentially leading to more refined AI systems capable of safe and meaningful human-computer interactions. Future research might explore the scalability of RLHF in training even larger models or its integration with other learning paradigms to further enhance performance and ethical compliance.

2.1.2 GPT 4

he latest iteration of OpenAI's language models, GPT-4, represents an unprecedented leap in scale and complexity in the field of artificial intelligence. While OpenAI has not disclosed specific details such as the exact number of parameters or the intricacies of its training strategy, it is speculated that GPT-4 may incorporate approximately 100 trillion parameters. This figure is often compared to the estimated number of synapses in the human brain, highlighting the model's vast and intricate network architecture.

Although the sheer number of parameters does not directly correlate with performance efficacy, GPT-4 demonstrates significantly improved outputs over its predecessors. The enhancements are reflected in the model's precision, depth, and reliability. GPT-4's performance has been rigorously evaluated through tests originally designed for human assessment, where it ranked in the upper decile of participants. This benchmarking underscores its advanced cognitive and processing capabilities.

A notable advancement in GPT-4 is its multimodal functionality. Unlike earlier versions focused solely on text, GPT-4 extends its competency to comprehend and generate outputs across multiple formats, including text, images, videos, and audio. This multimodality allows GPT-4 to engage in a more diverse array of applications, enhancing its utility in fields such as multimedia content generation, educational technology, and interactive entertainment.

Despite its advancements, GPT-4 is not without limitations. These include:

- **Hallucinations in Output**: GPT-4 sometimes "hallucinates" information, creating plausible but factually incorrect statements or endorsing false assertions made by users. This issue is a significant challenge in ensuring the reliability of its applications, particularly in domains requiring high factual accuracy.
- **Temporal Knowledge Constraints**: The training data for GPT-4 only extends up to September 2021, which limits the model's awareness of subsequent events or developments. This temporal gap can hinder the model's effectiveness in scenarios requiring up-to-date information.
- **Complex Problem Handling**: Similar to human cognitive processes, GPT-4 exhibits difficulties in solving complex problems, especially those involving intricate mathematical computations. This limitation reflects the inherent challenges in modeling higher-order cognitive tasks within a machine learning framework.
- **Persistent Biases**: Despite efforts to mitigate biases through techniques such as Reinforcement Learning with Human Feedback (RLHF), residual biases from the training data can still permeate the model's outputs. These biases can skew the model's decision-making processes and output generation, potentially reinforcing existing societal stereotypes.

2.2 Information Retrieval

Information Retrieval (IR) is a critical discipline within computer and information science, focusing on the extraction of relevant data from extensive collections to meet specific informational needs. This field involves the development and application of systems that locate and retrieve information resources such as text, documents, images, audio, and video from a structured collection of digital content.

An effective information retrieval system is structured around three fundamental components: indexing, document representation, and query modeling.

1. **Indexing**: This initial phase involves the processing and organization of documents to facilitate more efficient query operations. Effective indexing reduces the computational cost associated with searching large datasets by organizing the data in a manner that accelerates retrieval processes.
2. **Document Representation**: This component entails transforming documents into a format that can be easily processed by IR systems. Typically, this involves the creation of a condensed representation of each document that captures its most essential attributes, often using numerical or statistical models. Common techniques include the vector space model, where documents are represented as vectors of identifiers, such as term frequency-inverse document frequency (TF-IDF) weights.
3. **Query Modeling**: This process converts user queries into a standardized format that can be effectively matched against the indexed documents. Query modeling is crucial for interpreting the user's intent and transforming informal user input into a more structured form that can be compared against the document representations in the index.
4.

The concept central to user interaction with IR systems is the "information need," which represents the specific information that the user seeks. The retrieval process is initiated by the user entering a query, a formal statement of this information need, akin to search terms entered into web search engines.
In the realm of IR, queries do not identify a unique object within the collection but rather retrieve multiple objects that vary in relevance. This aspect is a significant departure from traditional SQL database queries, which typically return exact matches.

IR systems distinguish themselves from traditional database searches through the classification and ranking of results based on relevance. These systems employ various algorithms to compute a relevance score for each item in the collection, considering factors such as:

- The frequency and distribution of query terms within documents.
- The presence of query terms in strategic parts of the document like the title.
- The usage of synonyms and semantically related terms.
- Exclusion criteria based on predefined blacklists that filter out undesirable content.

The outcomes of an IR query are typically presented on a Search Engine Results Page (SERP), where the results are sorted by their relevance scores in descending order, from the most relevant to the least. This sorting allows users to quickly access the most pertinent information relative to their query.

2.2.1 Semantic Search

Semantic search represents an advanced paradigm in information retrieval, focusing on improving search accuracy by interpreting the deeper meanings—semantic content—of user queries and the corresponding data. This approach extends beyond traditional information retrieval methods by incorporating the principles of semantic understanding, primarily facilitated through the implementation of advanced techniques like word embeddings.

Semantic search seeks to refine the relevance and precision of search results by grasping the intent behind a user's query and the contextual meaning of terms within the data repository. This process involves analyzing the linguistic relationships and conceptual linkages between terms, allowing the search system to recognize and respond to the nuanced meanings of queries rather than just the literal text.

While keyword search methodologies excel in scenarios that involve single words or specific brand names—owing to their direct matching techniques—they often fall short in handling complex queries. In contrast, semantic search, powered by vector space models, demonstrates superior performance in several intricate search scenarios:

1. **Document Search**: Semantic search mechanisms excel in retrieving documents related to specific topics, even if the document's indexing does not directly align with the search terms used. This capability is particularly beneficial for academic or professional settings where the diversity of terminology can lead to missed results with keyword-only searches.
2. **Product Search**: In the context of e-commerce, semantic search facilitates the discovery of products that meet users' needs, regardless of whether the product descriptions explicitly match the search terms. This adaptability improves user experience by connecting shoppers with relevant products that might not have been discovered through traditional search methods.
3. **Information Search**: Semantic search enhances the ability to locate specific information within texts, documents, or websites without the necessity for exact keyword matches. This feature is invaluable for comprehensive research tasks and for users seeking detailed information on complex subjects.

The implementation of semantic search involves constructing a semantic understanding of text through machine learning models, such as word embeddings. These models generate vector representations of words that capture their meanings based on the contexts in which they appear. The semantic relationships between words are encoded in these vectors, allowing the search algorithm to assess the relevance of documents based on semantic closeness rather than mere keyword presence.

2.2.3 Retrieval Augmented Generation

To mitigate concerns regarding the generation of hallucinatory or unreliable outputs from advanced models like GPT-4, the implementation of a Retrieval-Augmented Generation (RAG) approach offers a promising solution. This method enhances the generative capabilities of question-and-answer (Q&A) systems by integrating a retrieval component that accesses a pre-defined document corpus to enrich the input provided to the QA system.

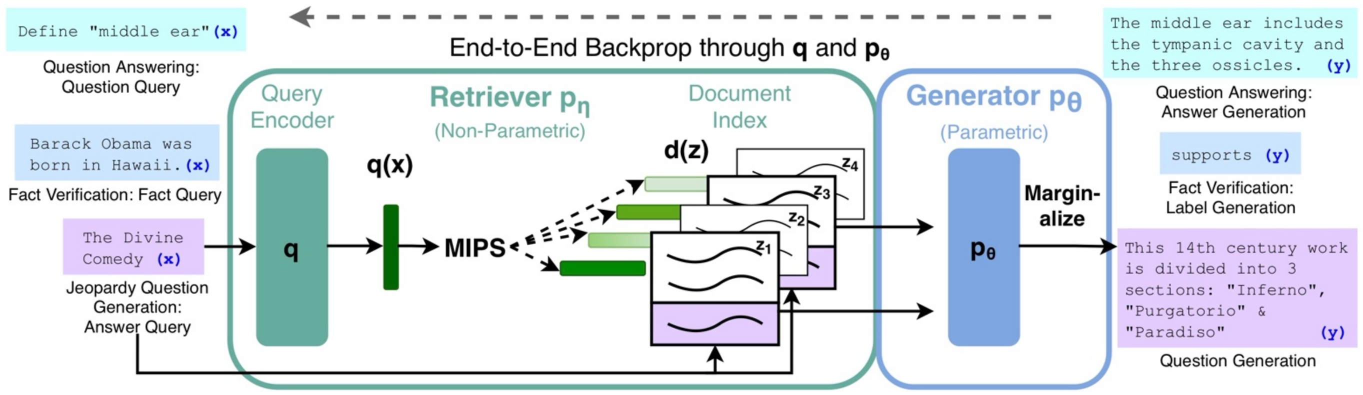

Define "middle ear" (x)
Question Answering: Question Query
Barack Obama was born in Hawaii. (x)
Fact Verification: Fact Query
The Divine Comedy (x)
Jeopardy Question Generation: Answer Query
End-to-End Backprop through q and pθ
Query Encoder
q
q(x)
Retriever pη
(Non-Parametric)
MIPS
Document Index
d(z)
z4
z3
z2
z1
Generator pθ
(Parametric)
pθ
Margin-alize
The middle ear includes the tympanic cavity and the three ossicles. (y)
Question Answering: Answer Generation
supports (y)
Fact Verification: Label Generation
This 14th century work is divided into 3 sections: "Inferno", "Purgatorio" & "Paradiso" (y)
Question Generation

Figure 6 Example of a Rag training approach

The core functionality of RAG can be conceptualized as an extension of the sequence-to-sequence (seq2seq) model paradigm, where the input sequence is transformed into an output sequence. The distinct feature of RAG lies in its initial step: prior to processing the input through the generative model, a retrieval subsystem is employed to identify and incorporate relevant information from an external dataset. This integration of non-parametric data (data outside the model's trained parameters) augments the intrinsic knowledge embedded within the model, thus enriching the output's relevance and accuracy.

The strategic inclusion of external knowledge sources into the generative framework enables dynamic updates and refinements to the informational content accessed by the model. By interfacing with comprehensive databases such as Wikipedia, the model leverages a vast repository of up-to-date and verifiable information, which significantly bolsters the model’s reliability and factual accuracy.

The architecture of a RAG system typically incorporates a pre-trained generative model to handle parametric knowledge, coupled with a pre-trained neural retriever that facilitates the extraction of pertinent information from external sources. The entire system undergoes a fine-tuning process, tailored to optimize performance across various dimensions, enhancing the model's ability to handle complex, knowledge-intensive tasks without the necessity for complete retraining.

This methodology not only elevates the model's performance by providing a mechanism for continuous verification of the generated content against reliable sources but also introduces a significant level of adaptability. The system can address new or evolving information needs through updates to its external data sources rather than through labor-intensive retraining of the entire model.

3. System Architecture and Model Integration

In this chapter, we examine the architecture of a sophisticated platform designed to streamline and enhance the retrieval of economic and financial information from the European Central Bank. This system leverages cutting-edge technologies to provide rapid, relevant responses to complex queries, improving efficiency for users needing detailed and accurate financial data. The architecture comprises three pivotal components:

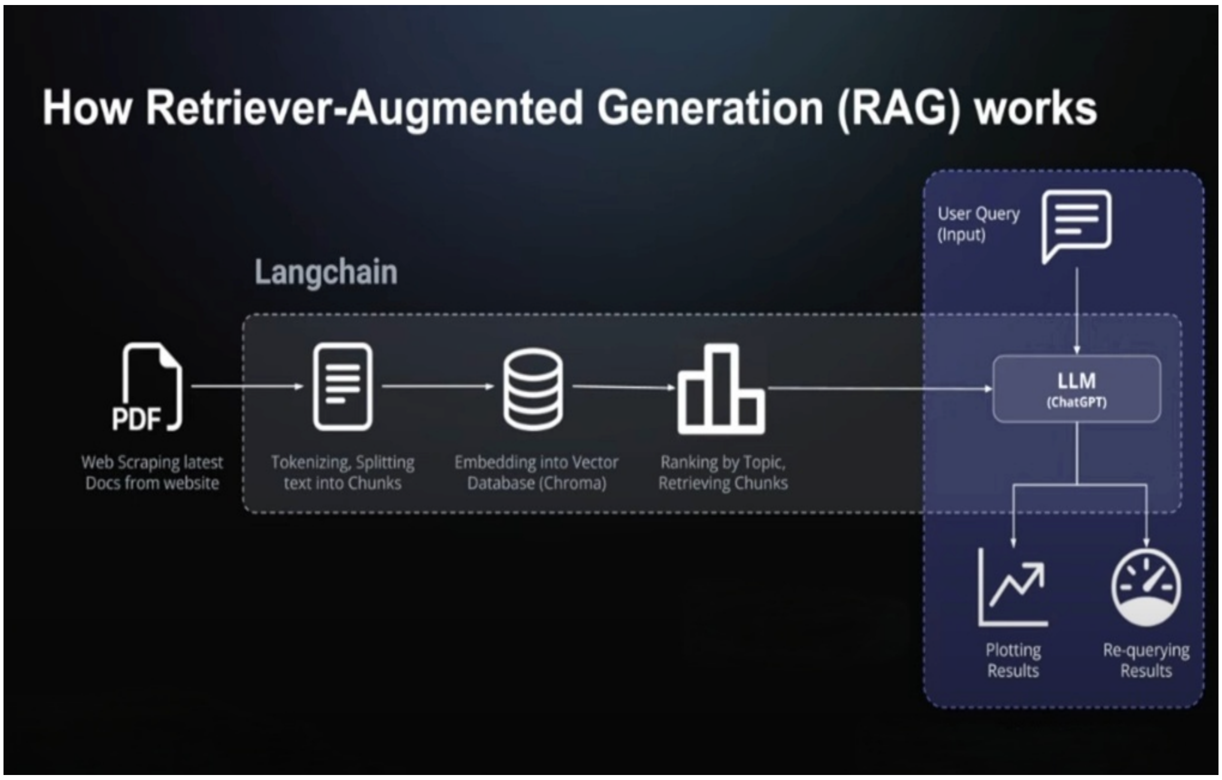
- **Knowledge Base and Data preprocessing**: This core component consists of an extensive collection of documents web-scraped from the European Central Bank's website. It includes various financial reports, policy documents, and other critical data. The library used for preprocessing this information is Llama Index, which utilizes several sophisticated techniques to enhance data handling and analysis. These techniques include Document tokenizing, Chunks Splitting and Text embedding into a Vector database

- **LLM and RAG Algorithm**: The system utilizes GPT-4, a highly advanced Large Language Model, capable of understanding and processing natural language queries. This model works in tandem with the Llama Index, a retrieval-augmented generation library, which enhances the LLM's response accuracy by incorporating relevant data extracted from the Knowledge Base during the generation process.

- **Web Interface**: Serving as the conduit between the user and the backend, the web interface is designed for simplicity and ease of use. Built using Dash, it facilitates real-time interactions, allowing users to submit queries and receive responses swiftly and efficiently.

The subsequent sections will provide an in-depth exploration of these components, elucidating how they interconnect to form a seamless and powerful tool for financial analysis and decision-making.

3.1 Application Workflow



How Retriever-Augmented Generation (RAG) works
User Query (Input)
Langchain
PDF
Web Scraping latest Docs from website
Tokenizing, Splitting text into Chunks
Embedding into Vector Database (Chroma)
Ranking by Topic, Retrieving Chunks
LLM (ChatGPT)
Plotting Results
Re-querying Results

Figure 7 Retrieval Augmented Generation Workflow

In the image above you can see how the application workflow is structured:

- **Web Scraping**: The process begins with scraping the latest documents from a designated website. These documents are typically in PDF format and contain relevant data needed for query responses.
- **Tokenizing and Splitting Text into Chunks**: After the documents are scraped, the text is tokenized, meaning it is broken down into smaller, manageable pieces or chunks. This step is crucial for processing the text in manageable segments.
- **Embedding into Vector Database (Chroma)**: Each chunk of text is then embedded into a vector database. This database transforms the text into a numerical format that machines can understand and process efficiently, using the Chroma tool for vector embedding.
- **Ranking by Topic, Retrieving Chunks**: The embedded text chunks are ranked by their relevance to specific topics. When a user query is made, the system retrieves the most relevant chunks based on the topic similarity to the query.
- **User Query (Input)**: This is the point of interaction where users input their queries into the system.
- **LLM (ChatGPT)**: The large language model, specifically ChatGPT, processes the user query. It utilizes the information retrieved from the ranked chunks to generate informed and contextually relevant responses.

- **Plotting Results**: The results of the query, once processed by the LLM, can be visualized or plotted. This step is useful for presenting the response in a user-friendly format.
- **Re-querying Results**: If further clarification or additional information is needed, the system can re-query the results. This is useful for refining the responses or diving deeper into specific topics.

In the subsequent chapters of this thesis, we will delve deeper into the most relevant parts of the workflow outlined above. Each chapter will focus on explaining key components in detail, providing a comprehensive understanding of how each segment contributes to the efficiency and effectiveness of the Retriever-Augmented Generation (RAG) model. This detailed exploration will include discussions on the methodologies used for tokenization, and data embedding, as well as the algorithms that drive the ranking and retrieval processes. Special emphasis will be placed on the integration of the large language model (LLM), specifically ChatGPT, highlighting its pivotal role in interpreting user queries and generating accurate, contextually relevant responses. By examining these components closely, we aim to illustrate the sophisticated interplay between advanced machine learning techniques and dynamic data retrieval systems that underpin the model's capability to deliver precise information swiftly.

3.2 Knowledge Base and Data Preprocessing

The knowledge base of the system is composed solely of documents from the Economic Bulletin because these documents are a rich source of current and authoritative economic and financial information. Choosing the Economic Bulletin as the exclusive source for the knowledge base has specific advantages, particularly in terms of content relevance and focus. The Economic Bulletin provides detailed analyses and reports on the economic conditions within the Eurozone, which is essential for users needing up-to-date European economic data. This focus ensures that the knowledge base is directly aligned with the needs of users looking for timely insights into economic and monetary developments.

Furthermore, by concentrating the knowledge base on a single, reliable source, the system can maintain high standards of data quality and consistency. This approach simplifies the maintenance of the knowledge base, as it minimizes the complexity associated with managing diverse formats and updating the data from multiple sources. It also enhances the efficiency of the query processing, enabling faster and more accurate retrieval of information, crucial for users who depend on swift data access for decision-making or analysis.

3.2.1 Document Ingestion

The documents from the Economic Bulletin are initially processed using the SimpleDirectory Loader of LlamaIndex. This tool is specifically designed to extract text from PDF files, converting them directly into string format. This step is crucial as it transforms the static, unstructured data contained in the PDFs into a structured, searchable format that can be efficiently indexed and queried.

3.2.2 Tokenize and split document into chunks

Following the initial extraction of text, the next step involves the tokenization and chunking of the content using the LlamaIndex Sentence Splitter. This process converts the continuous string of text into discrete tokens, which are essentially the basic units of text such as words and punctuation marks. After tokenization, the text is further split into manageable chunks—specifically, sentences in this case.

During the development process, several methods were explored to optimize the division of documents into manageable chunks. Some of the methods tested included:

1. **Paragraph-based splitting**: This technique divides the text based on paragraphs, preserving the inherent structure but often leading to uneven chunk sizes, with some paragraphs being too long and others too brief.
2. **Fixed-character count**: In this approach, the document is split into chunks containing a fixed number of characters. While this method ensures uniformity in chunk size, it often disrupts the natural flow of information by cutting sentences mid-way.
3. **N-gram segmentation**: This method involves breaking down the text into n-grams (a contiguous sequence of n items from the text). Although useful for statistical analysis, n-grams frequently strip context from the data, making them less effective for comprehensive understanding.

After evaluating these methods, the sentence-based division using the LlamaIndex Sentence Splitter was chosen. This approach was found to be the most effective because it maintains the semantic integrity of the information. Sentences typically represent complete thoughts or ideas, making them ideal units for processing. By using sentence-based chunks, the system can better understand and respond to queries, ensuring that the context is preserved and that the outputs remain high-quality and relevant. This method balances the need for manageable chunk sizes with the requirement to maintain sufficient contextual information for accurate data retrieval and analysis.

The use of LlamaIndex's Sentence Splitter is integral for breaking down extensive documents into smaller, coherent segments. This segmentation not only enhances the manageability of the data but also improves the effectiveness of the subsequent indexing and retrieval processes. By organizing the text into sentence-based chunks, the system can more accurately match user queries with the most relevant information, facilitating efficient and precise information retrieval.

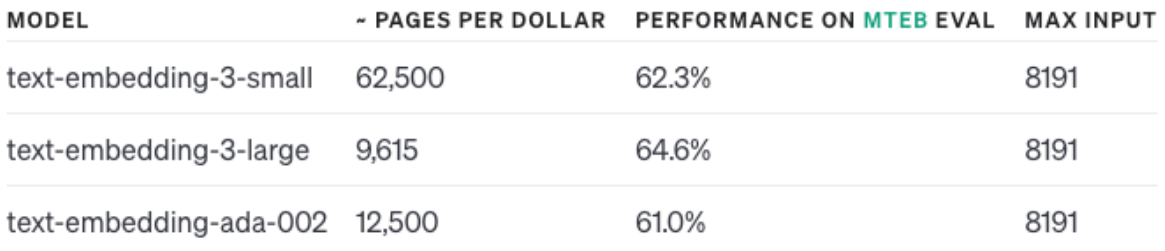
Conversion of tables to string format follows a different logic than plain text. To avoid producing unsatisfactory text that lacks semantic meaning and therefore goes unused, each cell in a row is provided with the corresponding field header. The cells are then combined with the other cells in that row, and multiple rows are merged until the token limit is reached. This helps ensure that the resulting text is effective and efficient.

3.2.3 Text embedder into Vector Database

The text embedding algorithm transforms the segments extracted by the Document Reader or the user's query into a vector format. Significant progress has been made in the last ten years to develop models capable of generating vector representations of words or phrases that encapsulate semantic relationships. This ability to represent semantic links is crucial for evaluating the relevance of a specific text segment in relation to the user's query.

The following criteria were taken into account when selecting the model to be employed:

- **Number of Input Tokens**: Choosing a model that can process a large number of input tokens is advantageous. This reduces the need for further text segmentation, enhancing the data storage efficiency in the database. It also speeds up search processes while ensuring high accuracy and relevance in the results.
- **Cost and Performance**: In addition to assessing the model's effectiveness, practical factors such as its cost, speed, and reliability were also examined. Striking a balance between cost and performance led to the choice of a model that offers the best value according to the project's needs. These are OpenAI embedding models considered while confronting the criteria mentioned above:



MODEL	~ PAGES PER DOLLAR	PERFORMANCE ON MTEB EVAL	MAX INPUT
text-embedding-3-small	62,500	62.3%	8191
text-embedding-3-large	9,615	64.6%	8191
text-embedding-ada-002	12,500	61.0%	8191

Figure 8 OpenAi Embedding models features

Text-emdedding-ada-002

Text-emdedding-ada-002 (TEA-002) is a general-purpose text embedding released by OpenAI in late 2022. This model can incorporate all the embedding models that OpenAI previously released and improve the performance in each task, such as search, similarity, and retrieval. In addition to enhancing performance, this model has expanded the input size and limited the output length, making the embeddings excellent for even very long text sequences while still achieving vectors with low dimensionality.

Since we are dealing with a closed model, details have yet to be released regarding the type of training and datasets used. The significant advantage of using this model is that it can be employed as a service, with dedicated APIs leading to zero infrastructure and maintenance costs. Furthermore, costs would be reduced over time, and there would be no dependence on potential malfunctions to OpenAI proprietary systems. The costs of this embedding model remain low at 0.00001€ per 1k token. If we estimate that a document page generally includes 8k tokens to reach 1€, we must convert 10.000 document pages. Furthermore, converting documents into vectors would only incur a one-time expense. The only recurring cost would be related to the conversion of user queries

Text-Embedding-3-Small (TES-003)

Text-Embedding-3-Small (TES-003) is an optimized text embedding model launched by OpenAI tailored for efficient processing with lower resource usage. Released in early 2023, TES-003 is designed to handle standard input lengths and produce concise vector outputs, ideal for applications requiring quick response times and moderate accuracy in tasks such as keyword search and basic document categorization. This model balances performance with lower computational demands, making it suitable for mobile and embedded applications.

Like TEA-002, TES-003 operates under a closed model framework where specifics about training methods and data are not disclosed. A key advantage of TES-003 is its integration as a lightweight service, which drastically reduces infrastructure overheads, as API access negates the need for local computational resources. Financially, TES-003 is priced competitively at 0.000005€ per 1k tokens, making it a cost-effective solution for developers needing frequent but less complex embeddings. The conversion of text to vectors incurs minimal one-time costs, with ongoing expenses primarily related to user query transformations.

Text-Embedding-3-Large (TEL-003)

Text-Embedding-3-Large (TEL-003), released by OpenAI in mid-2023, is a high-capacity text embedding model engineered for deep semantic analysis and complex query handling across extensive databases. This model supports exceptionally long input sequences and delivers high-dimensional vector outputs, enabling precise understanding in sophisticated applications such as legal document analysis, detailed content recommendation, and complex academic research retrieval.

TEL-003 continues to employ a closed model structure, with undisclosed training and dataset specifics. It offers significant advantages for enterprise-level applications, featuring robust API support that eliminates the need for extensive infrastructure and ongoing maintenance. The cost-effectiveness of TEL-003 is notable, priced at 0.00002€ per 1k tokens, with substantial discounts available for bulk usage. The conversion of documents into vector format is a one-time expense, and similar to TEA-002 and TES-003, the primary recurring cost is associated with processing user-generated queries.

3.2.4 Vector Database

Vector databases are specialized database systems designed to efficiently store, index, and retrieve vector data. Vector data, in this context, refers to the vector representations of various types of information, such as text, images, or audio, typically generated through machine learning models. These representations are high-dimensional vectors that encapsulate the essential characteristics of the data in a form that machines can process.

The primary advantage of vector databases is their ability to perform high-speed similarity searches among these vector representations. Using techniques like k-nearest neighbor (k-NN) searches, vector databases can quickly identify data points in the database that are most similar to a given query vector. This capability is crucial for applications in recommendation systems, image recognition, natural language processing, and other AI-driven technologies.

The principal advantages of Vector Databases are:

1. **Efficiency in Similarity Search**: They use advanced indexing techniques (e.g., tree-based, hashing-based) to speed up the retrieval of vectors that are similar to a query vector.
2. **Scalability**: Designed to handle large volumes of high-dimensional data, adapting well to both horizontal and vertical scaling.
3. **Flexibility**: Often compatible with various types of data inputs and machine learning models, allowing integration with existing AI/ML pipelines.

In this application ChromaDB was chosen as the Vector Databse to store the data. Chroma DB is a sophisticated vector database engineered to handle the complexities of storing and retrieving high-dimensional vector data efficiently. This type of database is particularly useful for applications that rely on fast similarity searches, such as those found in machine learning environments where embeddings generated from text, images, or audio need to be quickly compared and matched.

The core of ChromaDB's functionality lies in its architecture, which is designed to optimize both the scalability and speed necessary for processing large volumes of vector data. It employs a distributed system that facilitates parallel processing and effective data partitioning. This setup allows ChromaDB to manage vast datasets while maintaining high performance, a critical requirement for applications that demand real-time data retrieval.

One of the standout features of ChromaDB is its advanced indexing mechanism. Although specifics about the type of indexing might vary, typically systems like ChromaDB utilize space-partitioning data structures or hashing techniques to enhance the speed of vector retrieval. Such mechanisms are pivotal in reducing the time it takes to perform similarity searches, thereby improving the overall efficiency of the database.
Moreover, ChromaDB is built to integrate seamlessly with various AI models, particularly those used for generating vector embeddings. This compatibility is essential for maintaining fluidity in data processing pipelines, ensuring that data can be easily ingested from different sources and quickly turned into actionable insights.

In practical terms, ChromaDB can be an asset in numerous real-world applications, such as real-time recommendation systems or complex decision-making engines. For instance, in a recommendation system, ChromaDB could store embeddings of user preferences and product features, swiftly fetching the most relevant product suggestions based on user input. This capability not only enhances user experience through personalized recommendations but also supports dynamic updates to the database without sacrificing performance.

Additionally, the cost-effectiveness and performance optimization of ChromaDB make it a viable choice for businesses looking to implement powerful yet budget-conscious AI solutions. The database's ability to deliver high throughput and low latency is essential for maintaining user satisfaction, especially in services that users expect immediate responses.

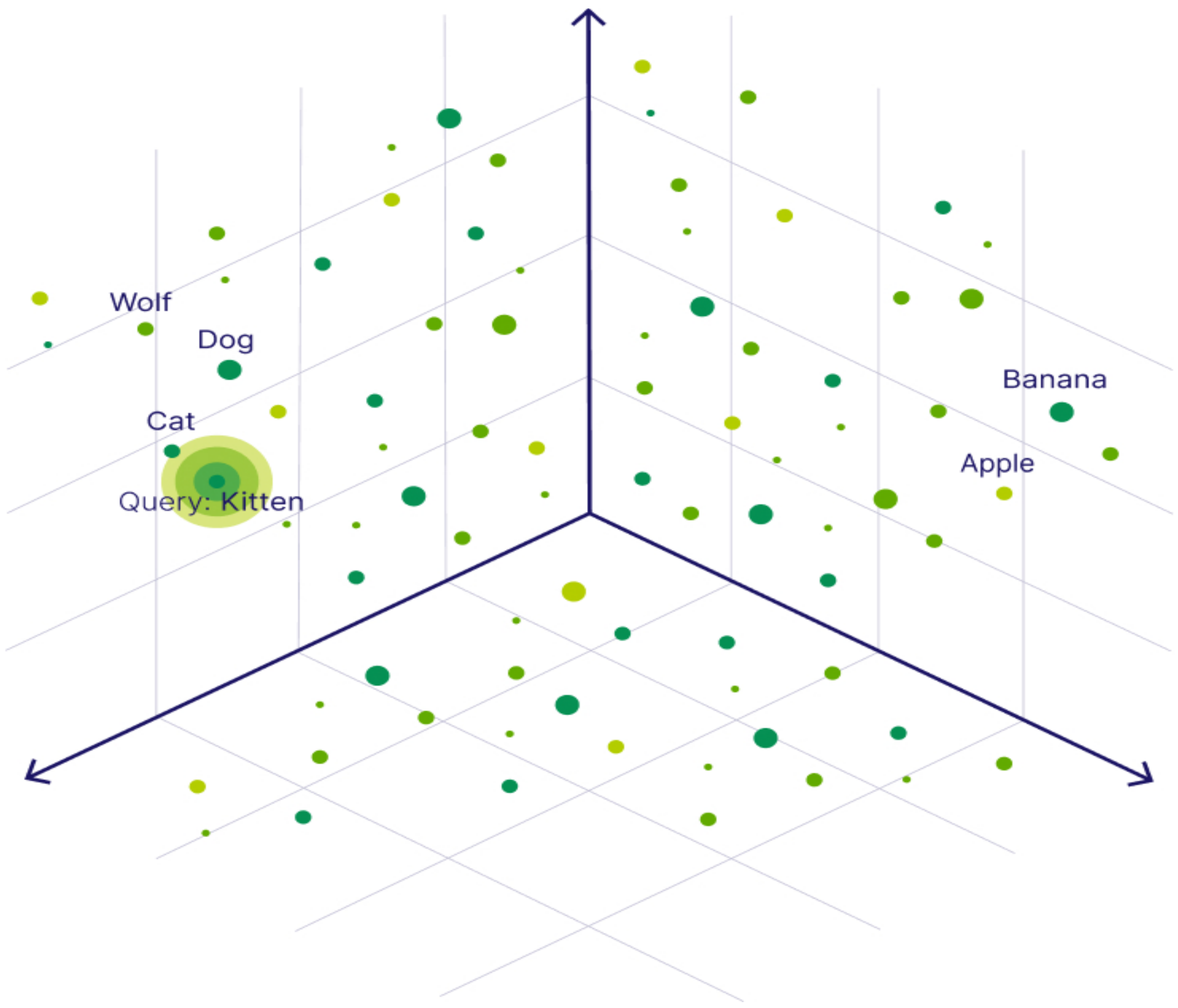


Figure 9 Graphical representation of a Vector Database

3.3 LLM and RAG Algorithm

Large Language Models, a subset of deep neural networks built primarily on the transformer architecture, have brought about a significant shift in the landscape of Natural Language Processing (NLP). Their unparalleled ability to comprehend, generate, and manipulate natural language has propelled the field forward. These models are trained using the technique of "next word prediction," where they are fed a sequence of words and tasked with predicting the subsequent word. Through this process, they acquire an understanding of the intricate relationships and statistical patterns inherent in human language.

The term "large" aptly describes these models, signifying both the vast amount of data used during training—often encompassing billions of words—and the complexity and size of the models themselves. Despite their computational demands, Large Language Models boast language processing capabilities that frequently rival, and sometimes even surpass, those of humans in terms of accuracy and fluency.

Their use in our application is focused on generating relevant responses based on text provided as context. Thanks to their deep language understanding, they can produce coherent and contextualized answers.

3.3.1 Fine tuning vs Retrieval Augmented Generation

Fine-tuning is a technique used to adapt pre-trained language models, like GPT, to perform specific tasks or better understand domain-specific data. Initially, these models are trained on massive amounts of text data in an unsupervised manner, which enables them to learn intricate patterns and relationships within language. Fine-tuning, however, tailors these generalized models to a particular task by updating their parameters with task-specific data.

The process of fine-tuning typically involves several steps:

1. **Pre-training**: Before fine-tuning, the language model undergoes pre-training on a vast corpus of text data. During this phase, the model learns to predict the next word in a sequence of text, capturing high-level language features and structures.
2. **Task-specific Data**: Fine-tuning begins with the selection of task-specific data. This dataset is labeled or annotated according to the desired task. For instance, if the task is sentiment analysis, the

dataset would contain text samples labeled with their corresponding sentiment (positive, negative, neutral).

3. **Fine-tuning Process**: The pre-trained model is then further trained on the task-specific data. During this phase, the parameters of the model are updated using techniques such as backpropagation and gradient descent to minimize the difference between the model's predictions and the ground truth labels in the task-specific dataset.
4. **Transfer Learning**: Fine-tuning leverages transfer learning, where knowledge acquired during pre-training is transferred to the target task. This reduces the amount of labeled data required for training and often leads to better performance, especially when the task-specific dataset is limited.

RAG (Retrieval-Augmented Generation), on the other hand, is a technique that combines the power of language generation models with information retrieval methods. Unlike traditional language generation models that generate responses solely based on the input prompt, RAG incorporates relevant information retrieved from an external knowledge source.

The RAG process is this:

1. **Knowledge Base**: RAG relies on an external knowledge base, which could be a large text corpus, structured database, or even a combination of both. This knowledge base contains factual information relevant to the domain or task.
2. **Information Retrieval**: When presented with an input prompt, RAG first retrieves relevant passages or documents from the knowledge base using information retrieval techniques. These retrieved pieces of information serve as additional context for generating the response.
3. **Generation Process**: With both the input prompt and the retrieved knowledge, the model generates a response that is not only coherent but also grounded in factual information from the external sources. This integration of retrieved knowledge enhances the relevance and accuracy of the generated responses, especially for tasks requiring access to external information, such as question answering or content generation.

For a question answering system based on organizational knowledge, a RAG (Retrieval-Augmented Generation) system is more fitting due to its dynamic access to evolving knowledge bases. In such a setting, where the information landscape is continually changing, RAG's ability to retrieve up-to-date and relevant information from dynamic knowledge sources aligns well with the task requirements. This ensures that the system can provide accurate and timely answers to queries, even as the organizational knowledge evolves.

Moreover, RAG tends to be less computationally expensive compared to some other approaches, making it feasible for real-time or large-scale deployment within organizational settings. By leveraging existing knowledge bases and integrating retrieved information into the generation process, RAG can efficiently generate contextually relevant responses without requiring extensive computational resources.

3.3.2 Retrieval of Similar Chunks in LlamaIndex

This chapter aims to elucidate the mechanism by which LlamaIndex, an information retrieval system, retrieves the most similar chunks based on user queries. The retrieval process is fundamental to the effectiveness of LlamaIndex in providing relevant information to users.
LlamaIndex employs a retrieval model based on the TF-IDF (Term Frequency-Inverse Document Frequency) algorithm, coupled with cosine similarity for measuring the similarity between documents and queries. This model enables LlamaIndex to rank documents according to their relevance to a given query.

TF-IDF Algorithm

The TF-IDF algorithm is at the core of LlamaIndex's retrieval model. It computes a numerical representation of each document and query based on the frequency of terms within them and across the entire document collection. The TF-IDF score of a term in a document is given by:

$$TF - IDF(t, d) = TF(t, d) \times IDF(t)$$

Where:

- $TF(t, d) TF(t, d)$ represents the term frequency of term tt in document dd.
- $IDF(t) IDF(t)$ denotes the inverse document frequency of term tt, calculated as $\log\left(\frac{N}{n_u}\right)$,, where N is the total number of documents and n_t is the number of documents containing term t.

Cosine Similarity

Cosine similarity is utilized to measure the similarity between the TF-IDF representations of documents and queries. It computes the cosine of the angle between two vectors, representing the documents and the query, in a high-dimensional space. The cosine similarity between a document d and a query q is given by:

$$\text{sim}(d, q) = \frac{\sum_{i=1}^{n} \text{TF-IDF}(t_i, d) \times \text{TF-IDF}(t_i, q)}{\sqrt{\sum_{i=1}^{n} \text{TF-IDF}(t_i, d)^2} \times \sqrt{\sum_{i=1}^{n} \text{TF-IDF}(t_i, q)^2}}$$

Where:

- t_i represents each term in the document and query.
- n denotes the total number of unique terms.

Retrieval Process

Given a user query q, LlamaIndex retrieves the most similar chunks by following these steps:

- **Tokenization and Preprocessing**: The query q undergoes tokenization and preprocessing to extract individual terms and remove stop words and punctuation.
- **TF-IDF Calculation**: For each term t_i in the query, LlamaIndex calculates its TF-IDF score in the query using the IDF values obtained during indexing.
- **Cosine Similarity Calculation**: LlamaIndex computes the cosine similarity between the TF-IDF representations of the query and each document in the index.
- **Ranking**: Documents are ranked based on their cosine similarity scores, with the most similar chunks ranked higher.
- **Retrieval**: Finally, LlamaIndex retrieves the top-ranked chunks as the most relevant ones to the user query.

3.4 Front End

In this chapter, we delve into the pivotal interface through which users interact with the advanced capabilities of our system.

The front end, constructed using the Dash framework for Python, serves as the linchpin that bridges the gap between the complex operations of language models and the end-users, who rely on this platform for querying and analyzing extensive documents, particularly those produced by central banks. This interface is not just a medium for input and output; it is also an embodiment of the application's usability, efficiency, and accessibility. By focusing on user experience design, we ensure that the application is not only functional but also engaging and easy to navigate, encouraging deeper interaction and exploration.

3.4.1 Dash Framework

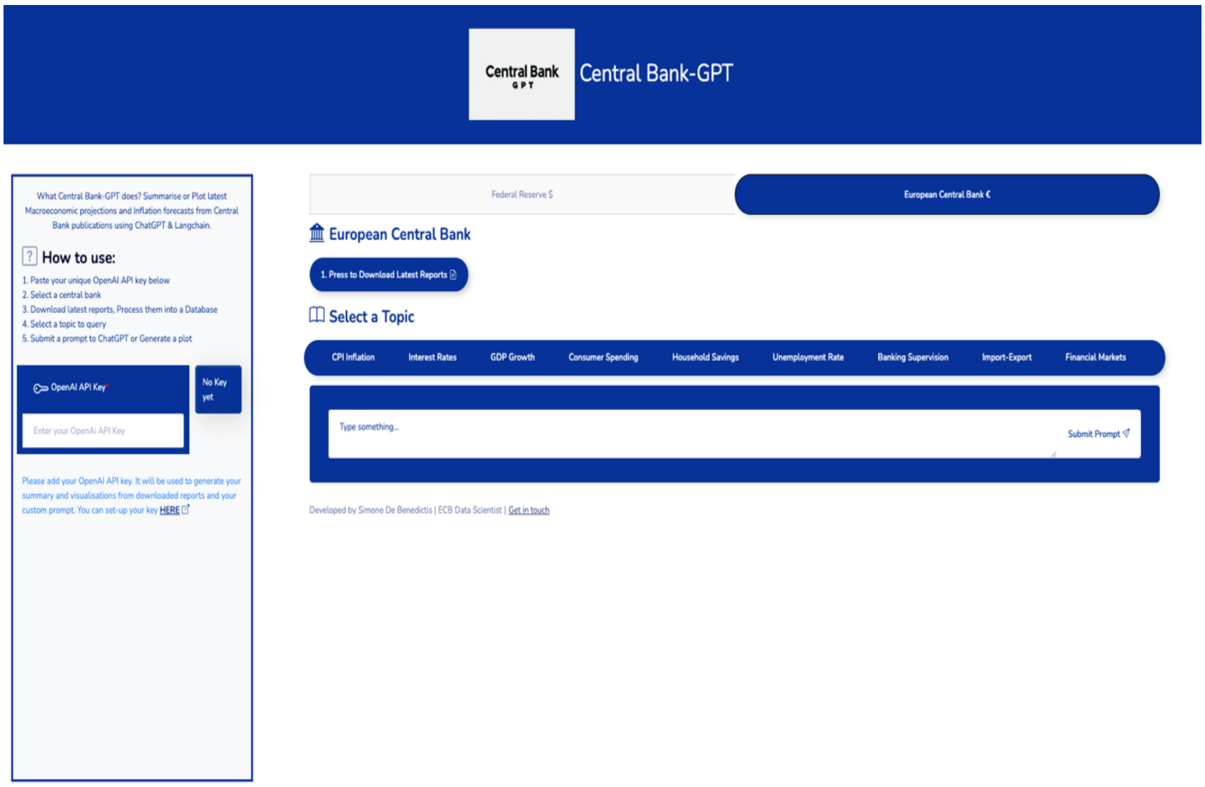
Dash is a Python framework for building interactive web applications easily and efficiently. Created by Plotly, it leverages the simplicity of Python to enable data scientists and analysts to develop complex, data-driven interfaces without deep knowledge of web technologies. Dash abstracts away the complexity of web development by using Plotly for high-quality visualizations and React.js for the front-end, making the UI responsive and modular.

Dash is known for its ease of use, with a component-based architecture that allows for interactive behaviors through simple Python callbacks. This means that interactions such as user inputs can dynamically update the content of the app without page reloads. It includes core components for layouts and can be extended with third-party components for enhanced functionality.

For deployment, Dash integrates seamlessly with Flask, allowing apps to be hosted on servers and accessed via URLs, facilitating both rapid prototyping and production deployment. Dash transforms the development of interactive web applications into a straightforward process focused on Python coding, making it a preferred tool for building professional-quality web apps in the data science community.

3.4.2 Application Front End

In the chapter dedicated to the front end of our Dash application, we explore the various components that collectively create a user-friendly interface designed for interacting with a large language model specialized in processing central bank documents. This interface is pivotal not only in guiding the user through the data retrieval process but also in providing a seamless and intuitive experience that encourages engagement and exploration.


What Central Bank-GPT does? Summarise or Plot latest Macroeconomic projections and Inflation forecasts from Central Bank publications using ChatGPT & Langchain.

How to use:

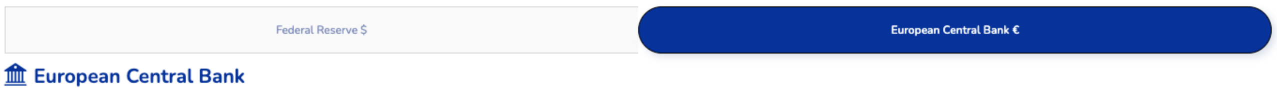
1. Paste your unique OpenAI API key below
2. Select a central bank
3. Download latest reports, Process them into a Database
4. Select a topic to query
5. Submit a prompt to ChatGPT or Generate a plot

Please add your OpenAI API key. It will be used to generate your summary and visualisations from downloaded reports and your custom prompt. You can set-up your key HERE

Developed by Simone De Benedictis | ECB Data Scientist | Get in touch

Figure 10 Front End view of the Dash APP

Bank Selection Section



This section of the interface features a selection tool where users can choose the central bank from which they wish to download documents. It provides a simple, streamlined method for accessing specific financial information by allowing users to select between different banks, such as the "European Central Bank" or the "Federal Reserve."

Download Button

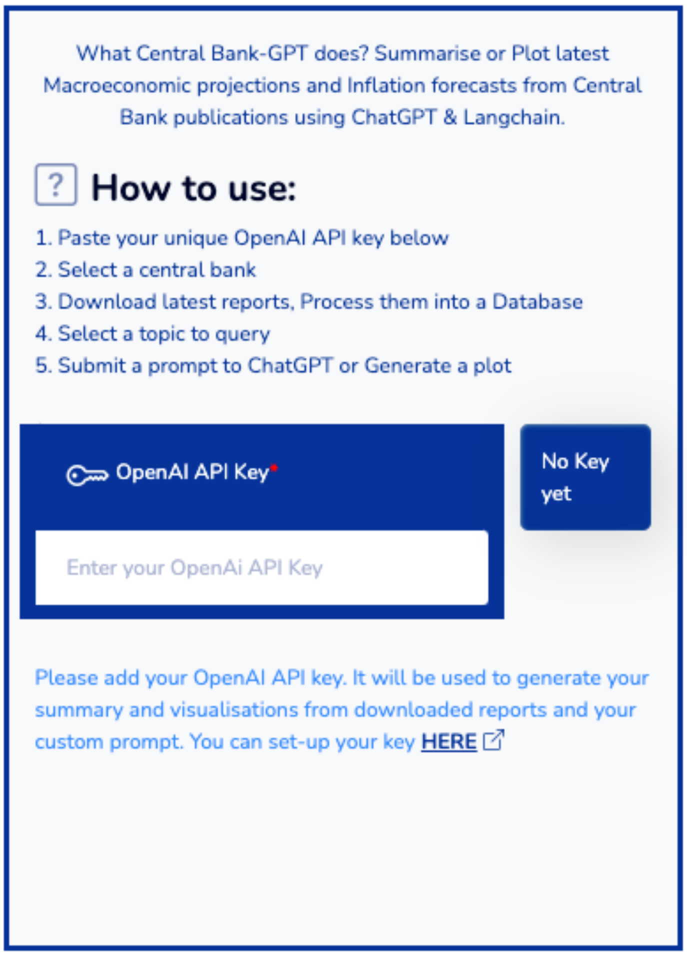
By pressing this button, the user can download the latest annual report relevant to the selected central bank. If the European Central Bank is chosen, the application will download the Economic Bulletin report for the previous year. Conversely, if the Federal Reserve is selected, it will retrieve the monetary policy reports from the past year directly from the Federal Reserve's website. This functionality simplifies the process of obtaining important financial documents, ensuring users have easy and immediate access to the latest data from these major financial institutions.

Vector DB creation Button

This button becomes active once the necessary documents are downloaded from the selected financial institution, such as the European Central Bank or the Federal Reserve.
When activated by the user, this button initiates a series of backend operations where the downloaded documents are tokenized and divided into manageable chunks. These chunks are then systematically ingested into a vector database. This process converts the textual content of the reports into vectorized formats that can be efficiently queried and analyzed through the application. The transformation into a vector database is essential for enabling advanced search capabilities and ensuring that users can retrieve specific information rapidly and accurately.

Sidebar/Help Section

This collapsible sidebar provides users with help regarding the usage of the application Giving instructions to how to leverage the OpenAi API Key to use the app.



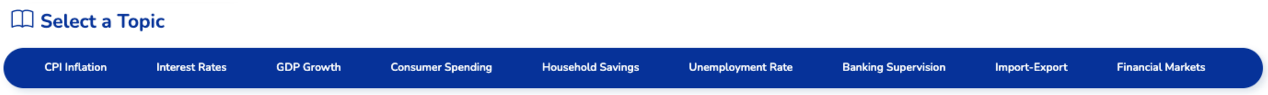
What Central Bank-GPT does? Summarise or Plot latest Macroeconomic projections and Inflation forecasts from Central Bank publications using ChatGPT & Langchain.

How to use:

1. Paste your unique OpenAI API key below
2. Select a central bank
3. Download latest reports, Process them into a Database
4. Select a topic to query
5. Submit a prompt to ChatGPT or Generate a plot

Please add your OpenAI API key. It will be used to generate your summary and visualisations from downloaded reports and your custom prompt. You can set-up your key HERE

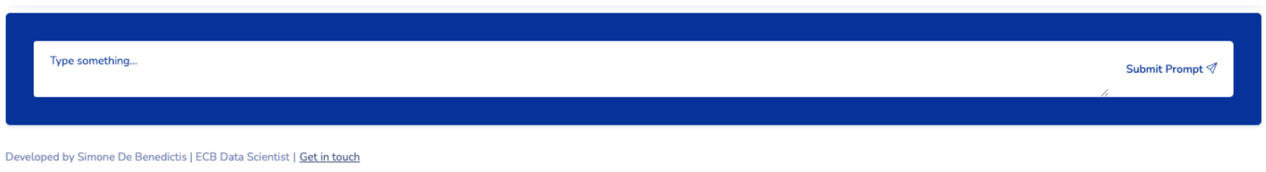
Topic Selection Section:



This section of the interface allows users to select specific topics, which tailor the displayed or queried information according to their needs. The available topics, such as "Inflation," "Interest Rate," and "Economic Growth," help refine the data retrieval process. Selecting a topic triggers a forecast prompt in the model, providing users with an overview of future trends related to that topic. It's crucial to select the appropriate topic before initiating other queries because each button not only filters the data but also supplements the query with additional keywords. These keywords aid in retrieving the most relevant chunks from the document database, ensuring that the information presented is precisely aligned with the user's interest. This functionality highlights the interface's role in enhancing the accessibility and effectiveness of the data retrieval process, making it integral to the user's research and analysis workflow.

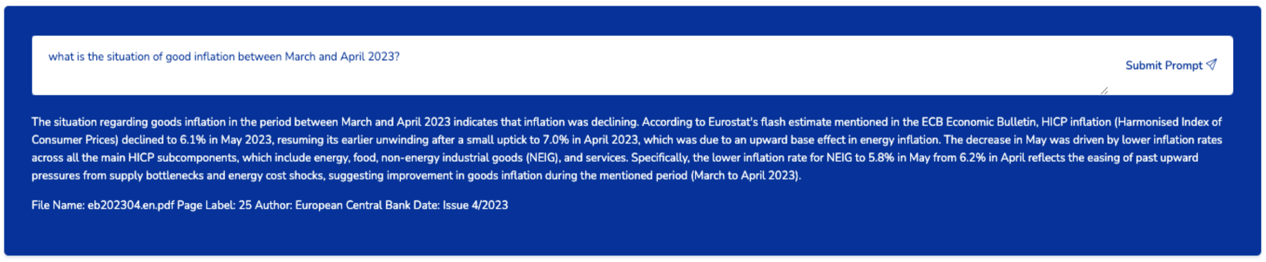
User Prompt Section

This section of the interface enables users to input customizable prompts. The responses to these prompts follow the chunks retrieval mechanism, where the model generates answers based solely on the information contained within the ingested documents. This approach helps prevent the model from producing hallucinated content, ensuring that the responses are reliable and directly reflect the data provided.


Type something...
Submit Prompt
Developed by Simone De Benedictis | ECB Data Scientist | Get in touch

Prompt example

In this chat example, the model responded to the user's prompt using information extracted from the documents we have ingested. Additionally, metadata such as the page number and the name of the document from which the information was retrieved are also provided. This allows the user to verify the accuracy and origin of the information furnished by the model, ensuring transparency and reliability in the data presented.



what is the situation of good inflation between March and April 2023?

The situation regarding goods inflation in the period between March and April 2023 indicates that inflation was declining. According to Eurostat's flash estimate mentioned in the ECB Economic Bulletin, HICP inflation (Harmonised Index of Consumer Prices) declined to 6.1% in May 2023, resuming its earlier unwinding after a small uptick to 7.0% in April 2023, which was due to an upward base effect in energy inflation. The decrease in May was driven by lower inflation rates across all the main HICP subcomponents, which include energy, food, non-energy industrial goods (NEIG), and services. Specifically, the lower inflation rate for NEIG to 5.8% in May from 6.2% in April reflects the easing of past upward pressures from supply bottlenecks and energy cost shocks, suggesting improvement in goods inflation during the mentioned period (March to April 2023).

File Name: eb202304.en.pdf Page Label: 25 Author: European Central Bank Date: Issue 4/2023

In conclusion, the front end of our Dash application represents a crucial interface between the user and the advanced technologies underlying the system. Designed meticulously to ensure usability and efficiency, it facilitates interactive and intuitive engagement with large language models tailored for processing central bank documents.

4. Conclusions

This thesis has presented the development and implementation of a Dash application designed to enhance the accessibility and interaction with language large models (LLMs) specifically fine-tuned with Retrieval-Augmented Generation (RAG) techniques. The system is tailored for querying and analyzing documents produced by central banks, leveraging the cutting-edge capabilities of natural language processing (NLP) and machine learning.

The theoretical underpinnings of NLP, including tokenization, vector space models, and deep learning architectures such as LSTMs and Transformers, provided a solid foundation for understanding the complex interactions within the application. The integration of Natural Language Generation (NLG) and the advanced iterations of the Generative Pre-trained Transformer models, particularly GPT-3.5 and GPT-4, emphasized the robustness of modern NLP applications.
The architecture of the application, focusing on the seamless integration of a knowledge base with a sophisticated text processing pipeline and a vector database, has demonstrated significant potential in enhancing the retrieval and interaction with specialized texts. The implementation of the RAG algorithm within this framework allowed for a dynamic and contextually aware response system that can adapt to the evolving needs and queries of its users.

Key findings from the deployment and testing of the application include improved query accuracy and user engagement. The use of semantic search and retrieval augmented techniques has shown to enhance the relevance and precision of the information retrieved, making it a valuable tool for researchers and analysts dealing with complex economic documents.
However, several challenges were encountered, including the scalability of the system with increasing data volumes and the continuous need to update the underlying models to accommodate new information and analytical techniques. Future work should therefore focus on enhancing the scalability and flexibility of the system. Additionally, further fine-tuning of the LLM for specific economic contexts and terminologies could improve the system's efficacy and user satisfaction.

In conclusion, this thesis has not only demonstrated the feasibility of integrating advanced NLP techniques into a user-friendly application but also highlighted the potential of such technologies to transform the way we interact with complex institutional documents. As the field of NLP continues to evolve, it is anticipated that applications like the one developed in this thesis will become increasingly pivotal in extracting meaningful insights from vast amounts of textual data.

4.1 Future Works

The research and development undertaken in this thesis have unveiled several opportunities for advancing the field and enhancing the implemented system. These opportunities span various aspects, including system design, model capabilities, and exploring new application domains, all of which are integral to the evolution of the application.

1) Enhancing System Scalability and Performance

One key area for future work involves enhancing the scalability and performance of the system. This includes implementing dynamic scaling mechanisms that effectively handle variable user loads and data volumes to ensure responsiveness. Additionally, optimizing the text processing and query execution pipelines to reduce latency and improve throughput could substantially benefit system performance, particularly with complex queries across extensive datasets.

2) Advanced Model Training and Fine-Tuning

Further refining the model's training and fine-tuning processes is essential. Introducing continual learning could enable the model to adapt to new data while retaining previously learned information, thereby maintaining relevance over time. Moreover, domain-specific tuning could significantly enhance the model's capability to understand and generate specialized economic content, improving the quality and applicability of responses.

3) Expansion of Knowledge Base

Expanding the system's knowledge base to include multilingual content would make the application more versatile and globally applicable. Additionally, integrating real-time data feeds would provide users with the most current information, a critical element in the dynamic field of financial analysis.

4) Text to SQL Integration

A promising direction for future work involves incorporating a Text to SQL (Structured Query Language) approach within the application. This approach would enable users to interact with the system using natural language queries that are automatically translated into SQL commands, facilitating direct interaction with underlying databases without the need for specialized database query knowledge.

The Text to SQL approach leverages natural language processing to understand and convert user inputs into SQL queries. By integrating this capability, the application can significantly enhance its accessibility and efficiency, allowing users to retrieve and manipulate data more intuitively. This functionality would be particularly beneficial in analyzing complex economic datasets where users may not always be proficient in SQL.

Bibliography

[1] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A neural probabilistic language model. Journal of Machine Learning Research, 3:1137–1155, 2003.

[2] T. Mikolov. Recurrent neural network based language model. Brno University of Technology, Johns Hopkins University, 20. 7. 2010.

[3] A. Graves. Generating Sequences With Recurrent Neural Networks. Department of Computer Science, University of Toronto, 2010.

[4] M. Daniluk, T. Rocktäschel, J. Welbl, & S. Riedel. Frustratingly Short Attention Spans in Neural Language Modeling. Department of Computer Science, University College London, 2017.

[5] Antoniak, M., & Mimno, D. (2018). Evaluating the Stability of Embedding-based Word Similarities. Transactions of the Association for Computational Linguistics, 6, 107–119.

[6] Antoniak, M., & Mimno, D. (2018). Evaluating the Stability of Embedding-based Word Similarities. Transactions of the Association for Computational Linguistics, 6, 107–119.

[7] Wendlandt, L., Kummerfeld, J. K., & Mihalcea, R. (2018). Factors Influencing the Surprising Instability of Word Embeddings. In Proceedings of the NAACL HLT 2018, 2092–2102.

[8] Vinyals, O., Toshev, A., Bengio, S., & Erhan, D. (2015). Show and Tell: A Neural Image Caption Generator.

[9] Hermann, K. M., et al. (2015). Teaching Machines to Read and Comprehend. Advances in Neural Information Processing Systems 28 (NIPS 2015).

[10] Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., & Wierstra, D. (2016). Matching Networks for One Shot Learning. Advances in Neural Information Processing Systems 29 (NIPS 2016).

[11] Vaswani, A., et al. (2017). "Attention is All You Need". Advances in Neural Information Processing Systems 30 (NIPS 2017)

[12] P. Bajaj, D. Campos, N. Craswell, L. Deng, J. Gao, X. Liu, R. Majumder, A. McNamara, B. Mitra, T. Nguyen, et al. Ms marco: A human generated machine reading comprehension dataset. arXiv preprint arXiv:1611.09268, 2016.

[13] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. Transactions of the association for computational linguistics, 5:135–146, 2017.

[14] N. Lambert, L. Castricato, L. von Werra, and A. Havrilla. Illustrating reinforcement learning from human feedback (rlhf). Hugging Face Blog, 2022. https://huggingface.co/blog/rlhf.

N. Kandpal, H. Deng, A. Roberts, E. Wallace, and C. Raffel, “Large language models struggle to learn long-tail knowledge,” in *International Conference on Machine Learning*. PMLR, 2023, pp. 15 696–15 707.

[15] Y. Zhang, Y. Li, L. Cui, D. Cai, L. Liu, T. Fu, X. Huang, E. Zhao, Y. Zhang, Y. Chen *et al.*, “Siren’s song in the ai ocean: A survey on hallucination in large language models,” *arXiv preprint arXiv:2309.01219*, 2023.

[16] D. Arora, A. Kini, S. R. Chowdhury, N. Natarajan, G. Sinha, and A. Sharma, “Gar-meets-rag paradigm for zero-shot information retrieval,” *arXiv preprint arXiv:2310.20158*, 2023.

[17] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel *et al.*, “Retrieval-augmented generation for knowledge-intensive nlp tasks,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020.

[18] S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. B. Van Den Driessche, J.-B. Lespiau, B. Damoc, A. Clark *et al.*, “Improving language models by retrieving from trillions of tokens,” in *International conference on machine learning*. PMLR, 2022, pp. 2206–2240.

[19] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, “Training language models to follow instructions with human feedback,” *Advances in neural information processing systems*, vol. 35, pp. 27 730–27 744, 2022.

[20]X. Ma, Y. Gong, P. He, H. Zhao, and N. Duan, “Query rewriting for retrieval-augmented large language models,” *arXiv preprint arXiv:2305.14283*, 2023.

List of Figures