AI Linguistics

Guosheng Zhang[1,2]

[1]Department of Computer Engineering, San Jose State University, San Jose, USA
[2]The Center for High Precision Patterning Science, Lawrence Berkeley National Laboratory, Berkeley, USA
guosheng.zhang@sjsu.edu

***Abstract*—The language of AI is embedding. To implement artificial general intelligence (AGI) and artificial super intelligence (ASI), we need to have a deeper understanding of embedding. This research mainly focuses on exploring the basic properties of embedding, including the semantic domain of embedding, the dimensionality of embedding, the operations of embedding, the database search of embedding, embedding reasoning, embedding augmenting, and other important aspects, with the aim of making a comprehensive observation of embedding, a language capable of continuous operation, and providing deep insights into the underlying design of AI architecture. It provides a basic methodology for transporting everything from the real world to the AI world and provides a broad reference framework for the development of AGI and ASI.**

***Index Terms*—AI architecture, AI linguistics, artificial general intelligence, artificial super intelligence, continuous operations, discrete operations, embedding completeness, embedding database, embedding dimensionality, embedding domain, general embedding, large language models**

I. Contents

II. Chapter 1, New Words and New Interfaces

AI operates continuously in a high-dimensional space, distinct from the discrete, lower-dimensional realm of human intelligence. This fundamental difference, coupled with the finite vocabulary of human language, creates a barrier to accurately conveying AI-generated concepts, contributing to the phenomenon of AI 'hallucinations' and potentially impeding future communication with advanced artificial super intelligence (ASI). To bridge this gap, we propose modifying Large Language Models (LLMs) to use continuous input and output by embeddings, liberating AI operations from the confines of human language and accelerating the evolution of AI. By leveraging FAISS and a translator for interpretation, we introduce a method to expand human vocabulary dynamically, based on proximity in embedding space. This approach aims to enhance the precision of human-AI communication, fosters clearer interactions with ASI, and facilitates the evolution of human language to better capture the nuances of AI thoughts.

A. Introduction

High-dimensional vectorized word operations are key to breakthroughs in large language models (LLMs), which also highlights the fundamental differences between AI thinking and human thinking. The primary distinction lies in the dimensional gap. Human thinking, constrained by the types of information transmitters between neurons [1], typically operates within a space of up to about 100 dimensions. The everyday thinking of most people usually does not exceed 10 dimensions. To engage in higher-dimensional thinking, we often compress part of this space, thereby reducing its dimensionality. For example, when considering the overall affairs of a university, we typically compress the information from various departments into a single-dimensional overview. Detailed consideration of a department's internal dimensions arises only when focusing on that department's internal affairs. This nested structure of thought [2] extends upwards to the Department of Education, state affairs, national affairs, and even global human affairs. Similarly, it stretches downwards to the curricula of individual professors and the management of each class, as well as horizontally to include inter-university organization and the coordination of external affairs. Human beings utilize this nested structure of thinking spaces to efficiently address the world's complex problems within a constrained dimensional space. This approach economizes thinking effort but introduces a significant issue: information distortion during transformation and transmission across different nested spaces, making it challenging to consider local nuances in broader contexts. In contrast, AI thinking can construct a higher-dimensional space through word embedding, enabling direct calculation of complex, global problems while attending to detailed local variables. Because there is no loss of information conversion in different spaces, AI thinking can be very precise.

Another fundamental difference is the mode of operation. The human thinking is largely based on language. Human language is made up of words, and a word is a tiny, defined area with fuzzy boundaries in the space of the human mind. The human thinking process involves jumping between different words in a discrete manner. Additionally, the vocabulary of human language is not evenly distributed within the mind space. When comparing various human languages, researchers found that different languages have differing numbers of words within a same domain. Similarly, within a single language, there are different numbers of words across various domains.

Humans create new words and eliminate those that are no longer needed, adapting their mental lexicon to the demands of reality [3]. The discrete mode of operation of the human mind requires that the results of human thinking must be settled on a series of words. In contrast, AI thinking uses high-dimensional vectors as tools to perform continuous operations. The result of its thinking can be any point in the high-dimensional space, but that point may not have an exact human word. Currently, large language models (LLMs) attempt to approximate these results by selecting the nearest word within this vector space. This discrepancy highlights a fundamental conflict between the limitless potential outcomes of AI thinking and the finite vocabulary of human language. Furthermore, this misalignment contributes to the phenomenon of AI 'hallucinations' in understanding and translation between AI-generated content and human languages. In each iteration where LLMs generate a word output, the generated approximate word is used as input for the next, causing errors to potentially accumulate [4] and the AI's otherwise very precise thinking to become more and more fuzzy and hallucinating.

The remainder of this chapter is structured as follows: the second part reviews human-computer communication research; the third part proposes our solutions; the fourth part discusses the simulation experiments; and the fifth parts concludes our findings.

B. Related Work

Enabling computers to understand human language in order to communicate effectively between humans and machines has been a central topic since the beginning of computers. Specifying a set of transistor circuit switching modes for each character like '01010010' is the low-level solution [5]. This scheme has been used to this day due to its simplicity, and it is estimated that it will not change until quantum computers use quantum spin schemes [6]. But machines can't really understand the meaning of these characters, let alone the meaning of words, sentences, and articles composed of these characters. So the initial effort to get machines to understand human language was just counting, comparing the number of times a word appears in two pieces of document to determine their similarity or 'meaning' or classification. 'Term frequency-inverse document frequency' (TF-IDF) is a masterpiece [7][8]. People pick out 'meaningful' words as 'features' for machine learning and count them to get a number if a piece of text has those words, otherwise get zero. These numbers become the characteristic values of a piece of text and are used for comparison and calculation. If we use keywords as feature names, we can also use TF-IDF to calculate the sparse vector of a text. For further refinement, this method was extended to the word level, which turned a word list into a one-hot matrix [9]. each word row only has a 1 at the intersection point with its own word column, and all the others are 0's. Such a matrix is an extremely sparse matrix. The next step is to evolve into a dense matrix with a greatly reduced dimensionality, which can be computed by 'principal component analysis' (PCA) or other feature engineering methods, where each word can be represented by a set of decimal vectors of lower dimensions. Word2Vec [10][11] greatly improves this process by using a neural network to calculate a vector value for each word based on a given corpus, using only one layer of neural network, which contains not only the meaning of the word, but also its context and syntax, as well as most of the information that the word presents in the given text. Because of its rich meanings and powerful functions, such a vector has far surpassed the previous PCA concept of laten features, and has evolved into a new species, so it is named word embedding, to distinguish it from all previous representations of words.

Word embedding is widely used to train language models [12][13], and it has shown amazing results because of its ability to understand the meaning of words in depth. Attempts to develop different embedding models [14][15] to help machines understand human language more deeply continued, and the embedding competition on Hugging Face has been very hot [16]. New embedding training models are constantly emerging, and few models stay in the first-place position for too long. In fact, this is a core competition for machines to understand human languages. The most powerful embedding models available today are already very complex large language models with more than billions of parameters [17].

In addition to developing better models, another effort is to change the way embedding is used from static to dynamic [18]. The trained embedding is used as the input of the language model, and then further trained during the language model training process to deeply fit the training text used. Dynamic embedding is currently the dominant use of generative AI models and is a key cornerstone of transformer architectures that can achieve amazing results [19]. After it was discovered that embedding and the hidden layers of language models have a substitution relationship [20], many models simply fused the two together, combining embedding models and language models into a supermodel with hundreds of billions or even trillions of parameters [21][22][23][24]. This simplifies the input and output of the models. In this way, it is hoped that artificial general intelligence (AGI) will be trained. When the training of static embedding generally adopts 1000-4000 dimensions [25], the large language model using multiple expert modules is highly integrated with the architecture of dynamic embedding, and its comprehensive dimension has reached 5000-20000 [26]. Because it has been found that the higher the dimension of embedding, the better the model's understanding of the language, and the stronger the model's capabilities [27]. We will dive into the relationship between the scaling laws and the embedding dimension in Chapter 3.

At the end of this path, we can expect AGIs that use higher-dimensional embedding, nest more expert modules inside, and have more complex structures and more parameters. For such an AGI, its output is still to select a suitable word from the human language vocabulary [28]. It may be more accurately selected than the current LLMs [29], and its knowledge boundaries can be closer to the boundaries of human knowledge. AGI is an encyclopedic generalist who has reached and approached the highest level of human beings in every professional direction and is the sum of all human knowledge. However, due to the limitation of input and output interfaces, the huge creativity of AI is locked by the limited words of human language, and the communication between AI and AI has become very difficult, because all current AI use human language as interfaces but human languages are foreign languages to AI.

C. Solution

The language of AI is embedding, and the language of humans is English or other natural languages. Here we see that a key bottleneck in the evolution from AGI to ASI is that it cannot break through the limitations of human language, and even if its AI mind creates new knowledge, it could be difficult for AI to accurately express its new concepts with the vocabulary of human languages.

1. New interfaces

To enable AI to overcome the constraints of human language, enabling it to think freely and communicate its ideas seamlessly with humans, we need to modify the language model's input and output to embeddings. This adaptation allows the AI model to operate entirely in its own language, from input all the way to output, facilitating learning, thinking, and communicating in a manner intrinsic to AI.

Then, we need a translator that is responsible for translating between human words and embeddings. When the AI model processes human language information, the translator converts this into embeddings. Upon generating thinking result, the model outputs it as embeddings too, which the translator then decodes back to human language, even creating new words if necessary. Therefore, we need a standard embedding dictionary. This dictionary is primarily for communication between humans and AI. That is because AI-to-AI communication can take place directly through their own language embedding. Currently, AI primarily learns and communicates using human-created knowledge. Therefore, if AI can adopt a shared dictionary to learn human knowledge, it also facilitates easier learning from one another, and allowing them to build upon each other's foundations.

Since embedding is a language that can be used for infinitely refined thinking, when all AI are free to use embedding, the universal language of the AI world, to learn, think and communicate, the output of one AI can become the input of another AI without barriers, and all AI can deeply cooperate to process more and more profound thinking and produce more and more profound insightful results, which will greatly accelerate the evolution of AI. It's like the reality of human experts to use the same language to communicate and stimulate new ideas and generate creative sparks for each other. Direct communication between AI allows AI to relay extremely fine thinking without stopping, translating embedding into human language only when it is necessary to communicate with humans. The current state is that AI models are reproductively isolated from each other and reproduce in solitary ways, and communication between AI requires the intervention of human engineers or human language tools to convey information, which is unnatural to AI.

2. New words

The first problem faced by translators for preparing human-AI communication is the asymmetry between embedding and human language vocabulary. Embedding has infinite possible outcomes, while human language has only a limited vocabulary. In the high-dimensional thinking space, an embedding of the AI thinking result may fall near a word in human language, and we can directly use this word to translate this embedding. This is common before artificial general intelligence (AGI), because the knowledge that AI has is basically all that can be expressed in human language. However, after artificial general intelligence, especially artificial super intelligence (ASI), it will continue to create some new knowledge, and some of the embeddings in its thinking results may fall in areas where there are no human language words nearby, and the translator will not be able to find suitable words to translate these embeddings. That's when we need to create new words to help humans precisely understand the idea of AI.

We can set a distance threshold for translating embeddings. If the translator finds a human language word within this spatial threshold, it translates the embedding into the nearest word. If no human language word falls within this threshold, a new word is created.

There can be a variety of ways to create new words to suit different specific situations. Here are three basic methods.

The first is the neighborhood method. Simply choose the 2-3 words that are closest to the embedding. This method is simple and straightforward and can be used in any situation. The way to combine them can be as simple as arranging the words together according to the distance from closest to farthest.

The second is the central neighborhood method. Select a top_k word combination, whose center point can reach the threshold, to form a new word. This method is a more accurate way to translate, as embedding B in Figure 1, can choose X, Y, and Z to form the new word.

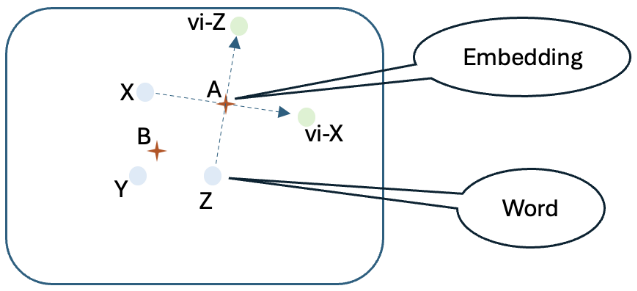

Fig. 1. Word formation

The third is the projection method. Sometimes, because all human language words are on one 'side' of the embedding, it is not possible to find a group of words whose center point is close to the embedding. For example, embedding A, at this time, we can first project the two closest words X and Z to the other side of A with A as the center, form two temporary auxiliary words vi-X, vi-Z, and then use these four words to form a new word, and then simplify it to X-Z-vi.

We can find some more ways to construct new words, but all of them should help people understand new words directly as much as possible, and grasp and use new words quickly. Human AI linguists and AI engineers can reshape and recreate new words based on their semantics after the translator creates them, making them more in line with people's language habits.

The embedding value of the new word is readily available, so the new word can be immediately put into the standard embedding dictionary and immediately put into use in the communication between humans and AI.

D. Simulation Experiments

The main purpose of our simulation experiments was to demonstrate an executable operational process, so only 30 English words were selected as an example of a mini language. Experiments that can demonstrate its practical application effects need to be carried out on a model that has basically achieved artificial general intelligence. Our simulation procedure is as follows:

1. Using the word usage frequency distribution data from the Wordfreq [29] library, 30 words were randomly selected from the 150 most frequently used words weighted according to their frequencies. These words were:

"the, to, and, of, a, in, i, is, for, that, you, this, at, by, your, an, all, they, like, about, what, who, their, there, she, new, other, its, these, work"

2. On Hugging Face, one of currently best open-source embedding model Salesforce/SFR-Embedding-Mistral [17] was selected to generate embeddings for these 30 words, dimension 4096, and a bidirectional translation dictionary was created as the basis of the translator. As a counterpoint, we also built a bidirectional translation dictionary of 100,000 words.

3. To build a phrase dataset, we randomly selected about 5B word sample documents from the large text databases COCA [48], Wikipedia Dumps [32], and American Stories [49], from which about 12800 sentences or sentence fragments, that contain only the selected 30 words, were selected.

4. Designed a GPT model, embedding in and embedding out. This GPT model used the overall architecture of 4 layers of 8-head encoders, and added position embedding to the input embedding, which was processed by the model and finally output embeddings.

5. Used the above phrase dataset to train the GPT model.

6. Based on the translation dictionary and FAISS [33] search, we designed a translator. We experimented with some prompts and translated some of the embedding outputs of the model, giving the translator a chance to generate new words by adjusting the parameter of the high-dimensional spatial distance threshold. At the same time, we used 100,000 embedding dictionaries to find the English words that this new word might correspond to. Some interesting examples are shown in Table I.

TABLE I
NEOLOGISMS COINED BY AI

	Example embeddings output by AI and translated for the mini language	***Embeddings translated with the 100k-word vocabulary***
1	and	cont'd
2	the-in	the-or
3	and-by-of	continued
4	and-by-vi	and-cont'd-vi
5	the-and-in	the-and-thew

As can be seen from the examples, if a suitable word cannot be found to translate the embedding in a 30-word mini-language system, there is still a big chance that a ready-made word cannot be found in a larger 100,000-word language system. This proves the extreme sparseness of human language in high-dimensional space. Large and small language systems often have some different translations of the same embedding, which explains some of the sources of AI hallucinations.

A good translator can continue to create new words in the process of accelerating the evolution of AI, ensure that humans can always precisely understand AI and communicate clearly with AI in both directions, and help humans achieve co-evolution with AI in intelligence.

E. Discussion

If we look at the communication between humans and other agents from the perspective of intelligence-centrism rather than anthropocentrism, it is inevitable that AI will use its own language, embedding, to learn, think, and communicate. And it's only natural that humans need a translator to communicate with AI. We propose dividing the large language model ecosystem into two parts. One is the AI model. AI models should be designed and optimized according to the embedding operation logic of the AI world, which requires us to conduct more in-depth research on the continuous computation in high-dimensional space. The second is the translator. The first translators were, of course, for the translations between human languages, such as English, and AI language embedding. In addition, there are many physical, chemical, and biological signals in the world that need to be translated into embeddings. In fact, what we need to do is embedding everything, completely translate our real world into the embedding world of AI, so that AI can truly understand the world and start scientific exploration in the real sense. Therefore, the development of various translators is an important prerequisite for the development of artificial general intelligence and artificial super intelligence. Imagine that if everything in our world has a corresponding existence in the embedding world of AI, AGI and ASI are a matter of course.

All translators need to be standardized, including the dictionaries and the distance thresholds.

F. Next Step

This chapter focuses on the two most fundamental aspects of AI linguistics, one is to unravel the linguistic limitations of

AI development, allowing AI to freely use its own language embedding to learn, think, and communicate. The second is to create new words to help humans understand AI more accurately, communicate and grow together with AI.

The accelerated development of AI has become inevitable, and how to maintain smooth communication between humans and AI has become increasingly urgent.

III. Chapter 2, Semantic Domain of Embedding

Embedding is converted from symbols, so its domain is not a point without volume. Based on this, we deeply explore the domain scope of embedding through various transformations from multiple perspectives, helping people to better understand the differences between embedding and ordinary vectors, and gain insight into AI language embedding.

A. Introduction

An embedding domain is the range of values that it keeps semantically unchanged. In a high-dimensional space, the space occupied by a word is a bounded region, in which all vectors have the same semantics, and when translated into natural language, they point to the same word.

In a high-dimensional space, the semantic boundaries of an embedding depend on its neighbors. We can search for its nearest neighbors in each dimension and calculate the intermediate value between itself and the neighbors in each dimension as its semantic boundary in that dimension. If our vocabulary has 100,000 words, and each word has 4096 dimensions, by sorting these 100,000 words in each dimension, then we can calculate the boundaries for each embedding.

Although it is ease to use Python to accurately describe the 4096 boundaries of an embedding, we still need to have an intuitive grasp of the scope of the semantic domain of an embedding. There are about 30,000 words in active use in English and Webster Dictionary contains about 470,000 words [30]. We combined the vocabulary of the University of Michigan[31] and Wikipedia dumps to select 300,000 words as the basic thesaurus for this experiment. Embeddings were then generated for these words using the open source embedding model Salesforce/SFR-Embedding-Mistral. We designed a translator whose search engine uses FAISS, mainly L2 and inner product (IP) search methods, to obtain the equivalent words of arbitrary embedding in English. In general, we set the distance threshold of the translator to no limit and use the nearest word in the embedding space as its corresponding English word.

The second part of this chapter will review the relevant research, and the third to sixth parts will explore the coefficients, random substitutions, inverse matrices, and eigenvector reconstruction matrices, respectively, to gain a numerical and intuitive understanding of the scale of the embedding semantic domain.

B. Related Work

The counterpart of embedding is a word in natural language, and a word in natural language space is a cloud of meaning with vague boundaries, rather than a geometric point without volume. Thus, we can see that an embedding is different from a vector in a mathematical sense. A vector represents a point with no volume, while embedding represents a cloud of meaning with a fuzzy boundary, so embedding has a certain volume. So far, research on embedding has focused on how to improve embedding training [34], evaluation, and application methods. In terms of the evaluation of embedding, the main focus is on the downstream application effect [35]. Exploration of the interior of embedding's semantic cloud is rare.

C. Change Modulus

Multiplying an embedding by a coefficient has a major effect on its modulus. We randomly selected 100 embeddings multiplied by a real number in the experiment, and the results of the experiment are summarized in Table II below.

TABLE II
Effect of modulus changes on the semantics of embedding

Multiplier k	***Effect***	***Remark***
$k < 0$	Semantic changes	
$0 < \|k\| < 1$	The semantic changes. As $\|k\|$ is lower than 0.005, the preferred meaning of all words stabilizes to 'rdquo'	The top 10 meanings of words from near to far: rdquo, hellip, ohioohio, snip, woodfin, gtgtgt, burnettes, brasfield, baldwins, buffington
$1 <= k <= 10^6$	The semantics remain the same	
$10^6 <= k <= 10^{18}$	Semantic changes, all words have their meanings shown as 'braconidae'	The top 10 meanings of words from near to far: braconidae, quack, mois, pmpn, undertake, unsanctioned, corporatised, moonwalks, eidoloclast, supersprint
$\|k\| > 10^{18}$	Semantic changes, all words have their meanings displayed as 'caird'	

It can be confirmed that when the range of k is [1, 1000,000], the size of the modulus has no effect on the semantics of embedding.

Considering that we are using L2 for searching, it is worth noting that at k is [1, 1000000], the range of embedding domain is very wide, because scanning the entire embedding space at 1000,000 times the length of this embedding modulus, the nearest embedding found is still the embedding itself. This also makes it clear that even some changes in the embedding direction will not affect its semantics.

The average absolute value of the dimension of embedding in our dictionary is 3.32, and within the range of the coefficient

[1, 10^6], the 'volume' of this semantically invariant spatial region can be calculated as a 4096D hypersphere.

$$V_n(r) = \frac{\pi^{n/2}}{\Gamma(n/2+1)} r^n \tag{1}$$

Here r = $3.32*10^6$, n = 4096.

So we get the approximate result of the calculation: 3.3198e+21834.

D. Change Direction

Changing the value of any one or more dimensions of an embedding will change its direction. According to the above experiments, a certain degree of change in the direction of embedding does not affect its semantics. So we randomly picked a few words and did some experiments to find the limit value of this acceptable angle change.

1. Change the value of any one of the dimensions

We arbitrarily change each dimension of an embedding individually in turn, and we get the following experimental results in Table III.

TABLE III
THE EFFECT OF ARBITRARILY CHANGING THE VALUE OF 1 DIMENSION

Change the value 'a' of any one dimension	***Embedding semantics***	***Ultimately stable semantics***	***Remark***
$\|a\| > 1000$	Change	'braconidae'	There are no key dimensions
$\|a\| <= 1000$	Not change		There are no key dimensions

If we change the value of only one dimension, the semantics of embedding will not change when the absolute value of the changed dimension is less than 1000. This shows that embedding is tilted within a certain range of any dimension axis without changing the semantics. The performance of all individual dimensions is the same, showing that there is no difference in weight between the dimensions in determining the semantics of embedding, and there is no so-called key dimension. When the absolute value of the changed dimension is greater than 1000, that is, when the angle of embedding tilting to a certain dimension axis is too large, the semantics of embedding will change and most of the changed semantics are rare words, misspells, markers, or personal names with low frequencies, such as 'gooden', 'appple', 'rdquo', 'brasfield', etc. As the absolute value of the dimension value increases, eventually all the semantics of embedding are displayed as 'braconidae', which is manifested as the phenomenon of hubness [36] The tolerance threshold of '1000' will also vary depending on the original value of this dimension value.

In contrast, the largest dimension value in our dictionary is 137.37656, the smallest is -120.926445, the average value of all dimensions is 0, and the average of their absolute values is 3.32. From this, we can calculate the angle change when a single dimension changes to 1000.

Using the dot product and magnitudes, the cosine of the angle θ between v and w is:

$$cos\theta = \frac{\mathbf{v} \cdot \mathbf{w}}{\|\mathbf{v}\| \|\mathbf{w}\|} \tag{2}$$

Finally, the angle θ is $arccos(cos\theta)$.

In this way, we calculated that the result is 77.11 degrees. That is, if the absolute value of arbitrarily changing a dimension is 1000, the direction of embedding will change by 77.11 degrees on average.

Since we can change the value of any one dimension, it is conceivable that in a space of 4096 dimensions, embedding can be tilted up to 77.11 degrees in any axis without causing a semantic change.

2. Change the value of multiple dimensions

In the case of keeping the absolute value of the dimension value less than 1000, we stratified sample 4 words as representatives among the 30000 most frequently used words, arbitrarily selected multiple dimensions and changed their values to [-10, 10] times of the original value, and obtained the average value of the experimental results in Table IV below.

TABLE IV
THE MINIMUM NUMBER OF DIMENSIONS THAT NEED TO BE CHANGED TO CHANGE THE EMBEDDING SEMANTICS

Multiplier k	***'the'***	***'river'***	***'aba'***	***'monolithic'***
-10	39.6	42.7	33.5	36.9
-8	49.1	42.9	51.5	44.0
-6	55.6	67.6	55.9	65.4
-4	66.9	125.8	137.6	83.0
-2	316.1	387.9	406.5	244.6
-1	703.0	752.3	891.1	555.7
-0.8	879.2	856.0	1061.3	755.5
-0.6	1077.4	1013.7	1296.6	913.1
-0.4	1310.7	1212.5	1610.2	1220.6
-0.2	1697.4	1498.3	2021.8	1622.1
-0.1	1928.3	1670.9	2294.6	1824.0
0	2231.2	1897.4	2607.1	2133.9
0.1	2636.6	2171.7	2997.8	2502.3
0.2	3186.2	2540.4	3515.0	2968.8
0.4	4096.0	3799.1	4096.0	4096.0
0.6	4096.0	4096.0	4096.0	4096.0
0.8	4096.0	4096.0	4096.0	4096.0
1	4096.0	4096.0	4096.0	4096.0
2	4096.0	4096.0	4096.0	4096.0
4	325.2	4096.0	1007.8	4096.0
6	70.0	3009.5	87.5	4096.0
8	54.7	423.0	65.9	1639.7
10	39.9	80.7	47.2	132.4

In Table IV, when k = -1, 'the' = 703.0, this means that if -1 is multiplied by the value of any certain dimensions in the embedding of 'the' to change its dimension values, on average, the values of 703 dimensions need to be changed for the semantics of the embedding to change. If the factor -1 is used to change the values of 702 dimensions, the semantics of embedding is still 'the'.

From the perspective of searching for neighbors, the values of the two adjacent embeddings will be relatively close in each dimension. If we take 2 times the absolute value as an estimate, the average values of the four words in the above table in the [-2, 2]interval are 2430, 2279, 2613, 2349 respectively. Compared with the total of 4096 dimensions, the two neighbors can be determined to be neighbors as long as the values of 1666, 1817, 1483, and 1747 dimensions are similar, that is, the average of 41% of the dimension values are similar.

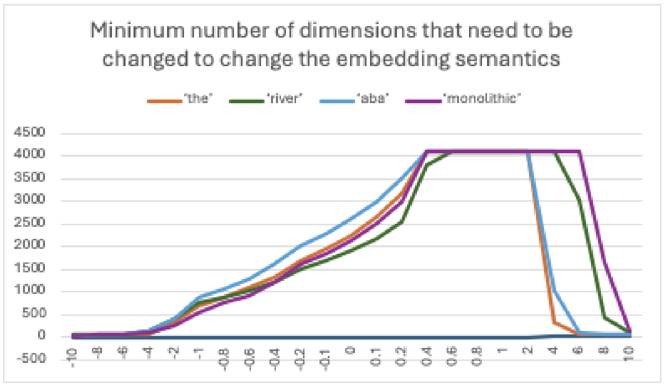

Fig. 2. Minimum number of dimensions that need to be changed to change the embedding semantics

As seen from Figure 2, when the multiplier factor is in the [0.4, 4] interval, the semantics of embedding do not change, no matter how many dimensions are changed with it. When the absolute value of the multiplier factor is greater than 10, the semantics of embedding will change as long as the value of a few dimensions is changed.

It is worth noting that when k = 0, the number of dimensions that can be changed reaches 2230, 1896, 2606, and 2132, respectively. In other words, in the 4096 dimensions of 'the' embedding, 2230 dimensions can be removed without changing its semantics. This is the semantic basis for which we can use the dropout strategy [37] when training a model.

These experiments show that embedding has a lot of redundant information and a large fault tolerance space, which reflects that the semantic domain of embedding is very large and will not be limited to a vector point without volume in the mathematical sense. From another point of view, it is also the result of the transformation of embedding from discrete symbols to continuous vectors, which reflects the characteristics of embedding's source - symbols, and may also reflect that the training of embedding is insufficient, and there is still a big room for further training. The process of embedding from symbol to vector is the process of AI language refinement.

It can be expected that in the downstream tasks using these embeddings, the training model will continue to refine the semantic domain of embedding, and the semantic domain of embedding will further adjust its boundaries and capture the semantics in the new training text to better adapt to the new specific task.

E. Moore-Penrose Pseudoinverse Matrices

The number of rows and columns of an embedding sequence is usually not equal, and in the general sense it does not have an inverse matrix. So we'll calculate its Moore-Penrose Pseudoinverse matrix[38]. The Moore-Penrose pseudoinverse of a matrix A is usually denoted as A^+. For any $m \times n$ matrix A, its pseudoinverse A^+ is an $n \times m$ matrix that satisfies the following four conditions, known as the Moore-Penrose conditions:

1. $AA^+A = A$
2. $A^+AA^+ = A^+$
3. $(AA^+)^T = AA^+$
4. $(A^+A)^T = A^+A$

The pseudoinverse A^+ can be computed using the Singular Value Decomposition (SVD) of A:

1. Singular Value Decomposition (SVD): Decompose A into three matrices:

$$A = U\Sigma V^T \tag{3}$$

Where U is an $m \times n$ orthogonal matrix, Σ is an $m \times n$ diagonal matrix with non-negative real numbers on the diagonal (singular values), and V is an $n \times n$ orthogonal matrix.

2. Construct the pseudoinverse of Σ:

$$\Sigma^+ = \text{diag}(\sigma_1^{-1}, \sigma_2^{-1}, \ldots, \sigma_r^{-1}, 0, \ldots, 0) \tag{4}$$

Where σ_i is the non-zero singular values of Σ.

3. Compute the pseudoinverse:

$$A^+ = V\Sigma^+U^T \tag{5}$$

We use stratified sample 30 embeddings to form a 30 $\times$ 4096 matrix, then use Python to calculate the Moore-Penrose inverse matrix of this sequence, and translate it into English, the experimental results are as following table V. Due to layout reason, only 9 samples are shown in the table.

The values in the inverse matrix are generally less than $10e-4$, and some are less than $10e-7$, which is not within the normal embedding value range. Multiply them by a coefficient $k = 110000$ to bring them all back to the normal embedding value range, and we get the same semantic meanings as the original matrix, although the new embedding has completely different values from the original embedding. The value of the coefficient k is in the range of $[1, 10^6]$, which is consistent with the experimental results in the previous section. After the inverse matrix transformation and appropriate amplification, it is still in the corresponding original embedding semantic domain.

F. PCA Reconstructing Matrix

Reconstructing a matrix through PCA allows us to focus on some of its features while keeping the basic properties of the matrix unchanged. Here we will use this feature of the SVD algorithm to reconstruct the embedding sequence in different ways to explore the breadth of its semantic domain.

TABLE V
ORIGINAL MATRIX AND ITS MOORE-PENROSE INVERSE MATRIX

Sample Words	***Original Embeddings***	***Inverse Embeddings * 110000***	***Semantic Meanings of Inverse Embeddings***
the	tensor([2.7491, 2.1026, 3.8282, ..., 7.5678, -4.4317, 2.5307])	tensor([-6.3303, 5.7066, 8.5918, ..., -0.6107, -2.9159, -11.1928])	the
river	tensor([8.9348, 5.8825, 1.5507, ..., 7.5117, -3.2387, 9.8582])	tensor([18.5763, 19.6233, -2.1471, ..., 1.6001, 2.2617, 15.7882])	river
industrial	tensor([1.9872, 0.8072, 4.5429, ..., 9.3002, -4.4041, 1.9959])	tensor([-0.2529, 6.8005, 4.7849, ..., -0.0566, 4.7662, -6.9177])	industrial
hanging	tensor([0.0170, -2.2427, -0.8381, ..., 10.3935, -4.9810, 4.9505])	tensor([-11.2721, -7.3838, -13.5687, ..., 10.1736, -0.8778, -5.0823])	hanging
bull	tensor([5.3079, 0.5402, 0.7918, ..., 5.9474, -4.4814, 4.7800])	tensor([12.5994, -0.8768, -10.0363, ..., -15.6166, 0.9301, -0.0330])	bull
advocate	tensor([5.1580, 0.2517, 0.4539, ..., 11.2693, -1.9855, 4.8887])	tensor([6.4504, -9.4736, -15.3024, ..., 10.1536, 6.3333, 10.4214])	advocate
promising	tensor([3.0843, -2.3224, 2.8488, ..., 11.2613, -3.4680, 7.4834])	tensor([1.5280, -7.1339, 3.6379, ..., 12.8480, 4.7336, 13.6657])	promising
creativity	tensor([0.4203, 0.6470, 5.5488, ..., 3.4081, -4.9273, -1.0625])	tensor([-10.4971, 1.4977, 6.2642, ..., -24.1196, 2.4547, -13.3778])	creativity
absorbed	tensor([1.3384, 0.0507, 4.8251, ..., 10.9183, -4.5461, 7.4510])	tensor([-9.9847, -5.6143, 14.7323, ..., 17.8744, 5.1543, 15.9429])	absorbed

Let's still construct a 30×4096 sequence matrix using the above 30 words as an example and calculate its U, Σ, and V by using $formula(3)$.

Then we try to change the value in Σ to construct a new embedding sequence.

Using $torch.svd()$, we compute eigenvalues:

tensor([1404.5420, 268.5848, 259.8046, 242.1995, 237.2298, 224.2604, 213.4968, 207.4843, 198.6976, 194.6998, 189.1008, 183.2787, 181.6937, 177.1088, 173.7043, 171.9798, 169.8064, 165.5437, 163.5864, 160.4978, 152.7189, 150.5603, 148.2517, 142.8485, 142.4684, 138.2381, 133.5845, 131.7041, 125.9783, 123.6764])

When we reconstruct a matrix with these eigenvalues, we need to use the first 19 eigenvalues to reconstruct a new matrix and get the same semantics as the original sequence. The combined weight of the top 19 eigenvalues is 76.43%.

We can also amplify the first eigenvector, for example, by multiplying it by a coefficient 10, and then reconstruct it to get a new matrix that is also semantic the same as the original sequence.

G. Discussion

Embedding's domain is very broad, which reflects the extremely sparse presence of natural language vocabulary in high-dimensional space. In other words, human natural language is very imprecise compared to the embedding language of AI. Much of the future trouble of communication between humans and AI roots in this, just like we are struggling to communicate human affairs management and quantum mechanics with our pets in English with hundreds of thousands of words, but our pets only have a few dozen words.

H. Next Step

AI and AI language embedding will continue to evolve, should humans let AI develop their own language embedding, or limit the evolution of embedding? Can humans limit the evolution of AI and AI language? These are profound questions that make people think deeply.

IV. CHAPTER 3, EMBEDDING DIMENSIONALITY AND AI ARCHITECTURE

How high a dimensionality does embedding need to perfectly reproduce all the knowledge in human natural language to help AI evolve to the level of AGI? which is currently difficult to measure directly. How many different embeddings or how many different expert modules do we need when we try to get the full knowledge in natural language by overlaying multiple embeddings or multiple expert modules? What are the special requirements for AI architecture for higher-dimensional embedding? In this chapter, we explore the mysteries of some mini language systems by taking a deep dive into them, and to pioneer the use of an engineering measurement method to initially answer these questions.

A. Introduction

If we need to completely reproduce an object in reality, a three-dimensional coordinate system is sufficient. But when we want to use embedding to completely reproduce a natural language in reality, we don't know how many dimensional coordinate system is needed. Because we don't know how many dimensions a natural language such as English has, we don't know exactly how many dimensions our AI model architecture can detect. The only thing we are currently comfortable with experimentally is Scaling Laws [27]. From the data of the original paper, it can be seen that when the number of layers of the model increases, the tail skewed in the figure will disappear. That is, it is to be expected that the parameters of embedding will also obey scaling laws if the number of layers increases correspondingly. Can we guess that if the dimensionality of embedding exceeds the actual dimensionality of English, is there still a marginal benefit of increasing the number of dimensions? This is like using 4-dimensional space to represent 3-dimensional objects, is the extra dimension still beneficial?

In Chapter 2 we can see that there are a lot of redundant dimensions in embedding. These dimensions may or may not make sense in the next model training. If we have a clear idea of how many dimensions English has, or how many dimensions our model can perceive in English, it will be very helpful for us to determine the number of effective dimensions of embedding, the number of layers in the model, and the number of epochs it is trained, because the number of layers of the model is closely related to the dimensions of embedding. Due to the high training cost of high-dimensional embedding, in the actual model design, we have adopted a multi-expert mode, which is the parallel use of multiple embedding systems from the perspective of embedding. The coverage of all dimensions of natural language by the Mixture of embedding / Mixture of expert (MoEm/MoEx) model is also worth exploring.

In the second part of this chapter, we will review the research on the embedding dimension, in the third part we will measure the engineering dimensions of some mini-language systems, and then in the fourth part we will estimate the engineering dimensions of English. In Part 5, we will explore ways to use low-dimensional multi-expert models and multi-embedding models to cover all dimensions of a language. We will discuss and summarize in Part VI.

B. Related Work

Severer et la. in [39] reviews the history of the word Embedding, since the emergence of the word Embedding, people have focused mainly on the structure of innovative design language models and hyponymy, morpheme, synonymy, etc., and few people pay attention to the ideal dimension setting, which is left to experimentation. There are a number of articles that study the embedding dimension setting of recommender systems [40][41][42], focusing on how to dynamically set the embedding dimension, and the dimensions they used in their experiments ranged from 20 to 220. Compared with large language models, recommender systems are less complex and do not have the similar depth and breadth.

Not only are there very few articles directly research the embedding dimension of language models, but the few studies that have taken the optimal Embedding dimension setting have also come to conflicting conclusions. Wang found in [43] that the optimal dimension setting for the large language model was around 200, and He et al. [44] verified that the optimal Embedding dimension setting should be 100-300, but they both set the upper limit of the highest dimension to 1000 in their experiments. OpenAI's lightweight model text-embedding-ada-002 published in the public documentation [45] has a dimension setting of 1536. This dimension is only 1/8 of their heavyweight Davinci-001 model, and Davinci-001 outperforms text-embedding-ada-002 text classification. That is to say, the davinci-001 model with about 12300 dimensions is better in text classification and certainly more expensive. The relevant embedding dimensions for each model of LLaMA 3 [46] announced are as following table VI:

TABLE VI
LLaMA 3 DIMENSIONALITY

	8B	***70B***	***405B***
Layers	32	80	126
Model Dimension	4096	8192	16384
FFN Dimension	6144	12288	20480

Their experiments proved that the higher the embedding dimension, the better the model performance.

This leaves a question mark over the optimal embedding dimension setting.

The experimental data obtained by Kaplan et al. [27] suggest that in the case of unlimited data and unlimited computing power, the training loss of large language models has the following relationship with the scale of model parameters:

$$L(N) = \left(\frac{N_c}{N}\right)^{\alpha N} \tag{6}$$

where $N_c = 8.8 * 1013$, $\alpha_N = 0.076$, N is the number of non-embedding parameters.

In other words, as the scale of model parameters continues to expand, the training loss of the model will become smaller and smaller and approach zero infinitely.

The dimensions of embedding have the power to substitute with the scale of the model's non-embedding parameters. In fact, embedding is also a neural network, and embeddings are the parameter weights of this neural network. In a dynamic embedding model, the embedding layer is a preface to the entire neural network model or can also appear in any other layer. So, while Kaplan's Scaling Laws focus on non-embedding parameter setting, in a neural network that includes dynamic embedding, embedding parameters are included in the total parameter size of the model. The difference between static embedding and dynamic embedding is only whether it is trained separately. From the perspective of embedding, we can even consider that the entire AI model is an evolutionary process of embedding. Based on this, we believe that in an AI model that includes dynamic embedding, the embedding layers and the other layers of the model are fused and intertwined. In Lan et la. [20] research, the substitution effect of embedding layers with other layers was confirmed. We will take Kaplan's Scaling Laws as a reference for embedding size research to some extent. The upper limit of the embedding dimension is the upper limit of the space dimension of the language itself, and we will try to find the upper limit of the space dimension of a language itself.

C. Measuring Engineering Dimensions of Mini Languages

The engineering dimension of language refers to the dimension that can be detected by a language engineering model. In this research, it specifically refers to how high-dimensional embedding is required for a standard GPT model to achieve a test loss smaller than epsilon.

The test loss here is the result of the detection of random samples of the training dataset after several training sessions, not the loss generated during the training process, nor is it a test result of another test sample. The reason for this design is that we want to test the learning ability of the model rather than its generalization ability. Moreover, many AI hallucinations do not appear because of insufficient generalization ability, but because they forget what has appeared in the training materials in the process of generalization. Therefore, in the whole detection process of our experiment, there is only one variable that is the dimensions of embedding. We are here to test the model's ability to learn using different dimensions of embedding. Ideally, if the trained dataset is a full knowledge base and a clean collection without noise, then we can expect the training loss to be the test loss. Essentially, with a basic amount of data, there is no overfitting without noise [47]. In the context of the full knowledge base, the knowledge in the downstream application is also part of the knowledge in the training set.

1.Build a mini language system

First, we use Wordfreq to randomly select 30 words from the 150 most frequently used words to form a thesaurus of the mini language. We construct a primitive library with 2B tokens composed of random samples of COCA, Wikipedia Dumps, and American Stories, and then extract sentences or sentence fragments composed of these 30 words from this primitive library to form a sentence library. In order for the sentence library to consistently reflect the real use of these words, our sentence search stop setting is as follows: if all 1000 consecutive searched sentences have been included in the sentence library, the search ends.

In the same way, we construct 5 mini language systems with 30, 40, 50, 60, and 70 English words.

2.Build a standard model

Our Standard Model is modeled after the self-attention mechanism[50] and OpenAI's GPT[51][52]]. To avoid dying $ReLU$ and guarantees smooth gradient flow, We also choose the Gaussian Error Linear Unit ($GELU$) as the activation function [53]. $GELU$ is defined as:

$$\text{GELU}(x) = x\Phi(x) \tag{7}$$

Where $\Phi(x)$ is the cumulative distribution function of the standard normal distribution:

$$\Phi(x) = \frac{1}{2}\left(1 + \text{erf}\left(\frac{x}{\sqrt{2}}\right)\right) \tag{8}$$

$GELU$ provides a smooth and continuous output, as opposed to the piecewise nature of $ReLU$, and $GELU$ can be approximated as:

$$\text{GELU}(x) \approx 0.5x\left(1 + \tanh\left(\sqrt{\frac{2}{\pi}}\left(x + 0.044715x^3\right)\right)\right) \tag{9}$$

This approximation makes it easier to compute in practice.

We randomly select a batch from the training set for each training session and evaluate it after every 100 batches to see if the training loss is less than the preset epsilon or if the training result is improved. The maximum number of steps set is 50,000 steps, patience = 100.

According to the cross-entropy formula:

$$H(p, q) = -\sum_i p_i \log(q_i) \tag{10}$$

Where p is the true probability distribution (one-hot encoded vector), q is the predicted probability distribution (output of the model's SoftMax function), i is indexes over the classes.

Because out of all p_i, only one is 1, and all the others are 0. Therefore, there is only one truly valid term in the cross-entropy formula, and that is the one correctly chosen. Considering that the value of each sample is between 0 and 1, assuming that the standard deviation is 0.1, for a normal distribution, the z-score corresponding to the 0.01 percentile is approximately -2.33, So to make 99% of the samples have a SoftMax probability greater than 0.5, we get the calculation:

$$0.5 = \mu - 2.33 \cdot \sigma$$

$$\mu = 0.5 + 0.233 = 0.733$$

In this way, the average probability value of the entire batch of samples in the correct option needs to be greater than 0.733, that is, the cross-entropy mean is less than 0.31 to ensure that the probability of generating an incorrect token is less than 1 percent, and if we relax the selection range to top_3, the probability of generating an incorrect token will be less than 1 in a million, so we can set epsilon = 0.31.

The other basic setup is to use a layer of decoder and a layer of feed forward architecture, using only one attention head, and the context length is 256 words, optimizer is Adam, learning rate = 1e-4, don't dropout [37] but use layer normalizations [54] and residual learning [55].

The results of the experiment are shown in Table VII below.

TABLE VII
THE MEASURED ENGINEERING DIMENSIONS OF MINI LANGUAGE SYSTEMS

Vocabulary Size	***The Size of the Sentence Library***	***The measured Engineering Dimensions (threshold=0.31)***
30	4239	82
40	11146	165
50	23855	312
60	46852	584
70	75910	945

D. Estimating Engineering Dimensions of English

Based on the experimental results in the previous section, take threshold = 0.31, we can build an exponential regression model as follows:

$$D = a \cdot \exp(b \cdot V) + c \tag{11}$$

where D is the engineering dimension, V is the vocabulary size.

The result after fitting is:

$$D = 43.1651 \cdot \exp(0.0455 \cdot V) - 95.1777 \tag{12}$$

Since the result is too large, we calculate the value of $ln(D)$. The number of words in Webster English dictionary is 470,000, and it is estimated that the engineering dimension of English is e^{21384}. If the average college student's vocabulary is 30,000, the engineering dimension of English actually used is about e^{1364}.

E. Multi-Embedding Architectures: MoEm & MoEx

In such a high-dimensional English space, if we want to use a single embedding to represent each word to achieve an error rate of less than one part per million, which is difficult to achieve with the current machine computing power, the natural choice is to use multiple lower-dimensional embeddings to depict a full-dimensional embedding. In practice, it can be a series of multi-embedding models, that is, multi-layer models that use residuals multiple times. It can also be parallel, i.e., a multi-expert model. Of course, it can also be a hybrid model that is sometimes connected in series and sometimes in parallel. Whether it is a series or parallel architecture, the essence is to use multiple lower-dimensional embeddings to cover a high-dimensional embedding space.

To calculate how many different embeddings are used in a model, we need to count each residual, and count each specialized module. Each feedforward layer can be roughly thought of as the end of an embedding module and the beginning of a new embedding module.

We designed two most basic experimental architectures, the series multi-embedding model (MoEm) and the parallel multi-expert model (MoEx), and the experimental results for 30- and 40-word mini-English are shown in Table VIII and Figure 3 below.

TABLE VIII
MOEM/MOEX EXPERIMENTAL RESULTS FOR 30/40-WORD LANGUAGE SYSTEMS

# of Embeddings/Experts	***30-Word Series***	***30-Word Parallel***	***40-Word Series***	***40-Word Parallel***
1	80	80	165	165
2	55	55	80	105
3	45	45	70	80
4	40	40	65	65
5	40	40	60	55
6	40	35	60	55
7	35	35	60	55
8	35	30	50	50
9	35	30	50	50
10	35	30	50	45
12	35	30	45	45
15	35	30	45	45
25	35	25	45	40
45	30	25	45	35
75	30	25	40	35
100	30	25	40	30
130	-	-	40	30
165	-	-	40	30
200	-	-	35	30

The values in the table VIII are the embedding dimensions required for the model to select top_3 for an error rate less than 1 part per million. The table shows that for a 30-word language system using a series model structure of 7 embeddings, it takes 35-dimensional embedding to achieve an error rate of less than 1 in a million.

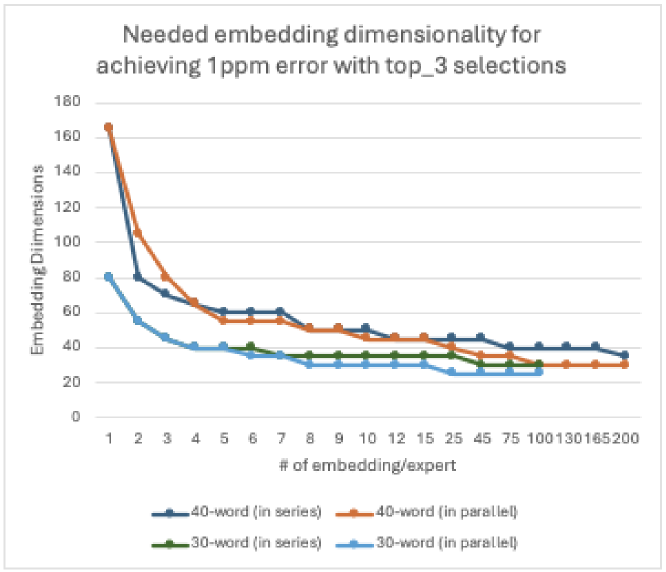

Fig. 3. Needed embedding dimensionality for achieving 1ppm error with top_3 selection

Here we can see that the 30-word language system has an elbow at 4 embedding/experts, and the embedding dimension is around 40. The 40-word language system has elbow at 5 embedding/experts, and the embedding dimension is around 60. The experimental results show that the marginal return of increasing the number of embeddings/experts is decreasing.

Adding the first embedding/expert almost halves the dimensions of embedding, then the rate of descent quickly slowed down. In these two language systems, the series model and the parallel model show a relatively consistent difference, that is, the parallel model can achieve the same training accuracy with a lower dimension of embedding. However, the need for dimensions in the series model declines a little faster at the beginning, and the elbow point appears a bit earlier. Therefore, combined with their respective advantages, it can be considered to do series connection first, and then parallel connection in the model design. That is, the number of layers of the model is increased first, and then the width of the model is expanded, and a certain number of expert models are connected in parallel.

According to the experimental setting, from the elbow points of these two samples, it is roughly estimated that for the practical application of 30000-word, to achieve the error rate of top_3 selection is less than 1 in 1 million, about 4000 embeddings/experts is needed, and the dimension requirement is about 40000.

Combined with some other AI model design techniques, we can further reduce the need for dimensions and number of embeddings.

F. Discussion

To further improve the research in this direction, the following aspects can be considered: First, the sample can be further amplified, increasing the duplicate sample threshold=1000 to 10000 or more, and more samples that meet the requirements can be collected from the large-scale data set. More samples may increase the engineering dimension of these simulated language systems.

The second is to deeply clean the sample, there will inevitably be some noise in the sample, and our experimental data was not cleaned because it was a simulation experiment. In real language model training, all data should be deep cleaned to prevent the model from learning the dimension of noise rather than the dimension of the language itself. Of course, language itself is also noisy, and what we need to clean is the noise outside of language. In other words, the engineering dimension of the cleaning sample may be slightly lower than our estimates.

Third, the distribution of logits is often skewed and heavily affects the SoftMax probability distribution. So we use the normal distribution here to analyze the distribution of probability, which is a simplified approximation, and it is necessary to extract the true probability value from the pre-trained LLMs in large quantities, and an empirical research can obtain the actual probability distribution to achieve more accurate results.

Fourth, if the number of training steps is extended indefinitely, the same training accuracy can also be achieved with lower-dimensional embedding. It takes a long training journey to determine the extreme training accuracy at a dimensional setting, so we determined that patience = 100. Of course, we can also expand the batch size, and use more fine-tuning techniques such as batch normalization and dropout to increase the stability of training, which could reduce the need for training steps.

G. Next Step

In this study, we did not use complex nesting structures, such as series structures nesting in parallel or parallel structures nesting in series. The multi-layered nested structure of extensive connectivity is the basic structure of human natural language. The current model architecture of AI is still in the simple stage, especially since there is little communication between the various components. Based on the two basic structures of series and parallel, strengthening the design of nested modules with different functions, strengthening the functional division of each module within the model, and strengthening the extensive connectivity between modules are the future evolution direction of AI model architecture.

V. Chapter 4, Embedding Database

The rapidly expanding embedding database presents a challenge for the search business. The existing search methods are excellent, but they are still far from the ideal of O(1). This chapter will focus on the development of a new search algorithm and embedding database architecture, which is a key step towards the eventual realization of searching arbitrary embedding in databases of any size at O(1) time complexity. Based on the research results in Chapter 2, we innovate the search algorithm of embedding. One is to sort all embeddings in each dimension and establish a semantic domain for each embedding. The second is to train a small AI model for each dimension to achieve O(1) search in a single dimension. Thirdly, according to the principle of dimension redundancy of embedding, combined with statistical rules and semantic domain principle, the search time complexity of O(5) is realized on a single machine.

A. Introduction

According to our research in Chapter 2, embedding is a huge cloud of meaning as the semantic domain, so the search of embedding is actually a search of the cloud of meaning rather than a point search in the mathematical sense. Embedding's matching is a matching of meaning clouds, not a matching of point locations. In the past, the design of vector database algorithm was basically based on the principle of point position matching, and the nearest neighbor search was implemented by calculating its distance. The methods for calculating distances are mainly Euclidean and cosine similarity. Various tree search algorithms take long paths from roots to leaves. These methods are not designed according to the actual characteristics of embedding, but according to the mathematical vector points of the high-dimensional space, so the efficiency is low, and it is difficult to achieve the ultimate goal of O(1) in theory. With the vigorous development of AI and the explosive growth of embedding database scale, embedding database search latency will constitute the most important response delay in AI applications.

This chapter will explore a novel embedding database architecture and search algorithm to achieve O(5) time complexity, especially because it facilitates distributed search deployments and can immediately implement O(1).

We will review the relevant vector database techniques in the second part, propose our O(1) solution in the third part, conduct the relevant experiments in the fourth part, and then discuss the next steps.

B. Related Work

At present, the vector search algorithms[56] in addition to the brutal force, mainly include various trees, including KD-tree [57], Ball-tree [58], R-tree [59], and M-tree [60]. The main purpose of these algorithms is to achieve accurate search. In addition to exact search, there are some approximate search algorithms, mainly various hash algorithms based on grouping[61][62] and sample tree search[63]. Graphical search algorithms[64] and compressed search algorithms[65] have also been applied in some specialized fields.

Due to the widespread use of distributed computing and hardware optimization, it has become practical to apply brutal force search even in large databases. For example, indexFlatL2 and indexFlatIP in FAISS [34] have good performance. However, in the face of very large databases, the time cost of brutal force search will be high.

From the perspective of the core algorithm of various real-world application search algorithms, KD-tree is a classic binary tree search, and Ball-tree mainly changes the node in the binary tree from a point to a ball, that is, a group of points. R-tree further changes the ball to a rectangle. M-tree further expands the scope of node to be arbitrarily defined, making it adaptable to any application scenario. Therefore, the evolution process of tree search is mainly the evolution of the definition of node in the tree, but this cannot change the essence of tree search is a hierarchical multi-step search process.

Graph search can essentially be seen as a variant of tree search, but with better adaptability to some naturally formed network structures, such as network of friends, roads, biosphere, etc. The compression algorithm uses dimension splitting or reducing data accuracy as the basic method to achieve faster speed, which has its application advantages in some cases. However, the basic algorithms behind this arrangement are still tree search, which are various specific applications of tree search and improvements for specific application objects.

C. Solution

Our proposed search algorithm is Domain Search. Take the embedding database with dimension D as an example:

1. Initialize the database

The sorting of high-dimensional vectors is nonlinear, so we need to simplify the sorting process by sorting all the data by each dimension. This way we can get D sequences. In each sequence, we calculate the semantic domain of each vector, that is, the middle dividing point between it and its upper and lower neighbors in this dimension, which is also the starting and ending values of its semantic domain in each dimension. We just need to calculate the average of its values with its upper and lower neighbors in this dimension. In each dimension, we form a sequence of the starting values of the semantic domain of all vectors, and each element in the sequence includes the starting value of its semantic domain, the end value, and its embedding.

2. Establish D AI models

The user submits a query vector, and the database needs to find its nearest neighbors in the sequence of each dimension. Direct method can be used is binary search, tree search, but they cannot reach O(1). Therefore, we design a small AI model here, and sample the data of each dimension series to train this AI model, each dimension series has a different distribution, so we need to train a dedicated AI model for each dimension. For example, for the first dimension value of 1.024 of a query vector, we can use the AI dedicated to the first dimension to predict the semantic domain to which it belongs at one time, to achieve O(1) in a single dimension search. Of course, after the AI predicts its semantic domain vector, we still need to do a verification based on the upper and lower bounds of the semantic domain. If not, we can determine whether to search up or down linearly to find the correct semantic domain vector based on the upper and lower bounds of this semantic domain.

3. Search

Since embedding is a cloud of meaning, searching for embedding's neighbors does not require checking all dimensions. According to the calculations in Chapter 2, we only need to match in 41% of the dimensions to presume that they are close neighbors. According to the laws of statistics, among the 4096 dimensions, we randomly sample several dimensions to verify whether they match, and then presume whether they are neighbors.

If we assume confidence = 0.99, then $Z\_score = 2.576$, we know that the required proportion is $p_0 = 0.41$, and assuming that the proportion of matches observed by the sample is $\hat{p}$, then the required sample size n is in the following formula:

$$Z = \frac{\hat{p} - p_0}{\sqrt{\frac{p_0(1-p_0)}{n}}} \tag{13}$$

Substituting the relevant values, if the first few dimensions of the sample match exactly, we get $n = 5$. After finite population correction (FPC) adjustment, the result is still 5. That is, if the first few samples match exactly, only 5 samples are needed to determine that it is a neighbor. In the case of different sample matching ratios, the number of samples required is as following table IX:

In this way, we can search the 4096-dimensional embedding database with only 5 dimensional searches to lock the target and complete the search.

4. Add data

When adding a new embedding to the database, we first find the semantic domain vector in each dimension series according to the search method above and re-divide the semantic domain of the three neighboring vectors according to the database

TABLE IX
NUMBER OF SAMPLES TO DETERMINE NEIGHBORS

Proportion of Dimensions that Sample Matches	*Number of Samples Needed to Determine Neighbors (4096D)*	*Number of Samples Needed to Determine Neighbors (any D)*
1.00	5	5
0.95	6	6
0.90	7	7
0.85	9	9
0.80	11	11
0.75	14	14
0.70	20	20
0.65	28	28
0.60	45	45
0.55	81	82
0.50	190	199
0.45	807	1004

initialization method to complete the data addition. Then, the AI models need to be retrained in each dimension.

5. Delete data

To delete an embedding from the database, we also follow the above search method to find the embedding and its upper and lower neighbors in each dimension sequence, and then merge its semantic domain with its adjacent domains and reassign it to its upper and lower neighbors. Then, the AI models need to be retrained in each dimension.

6. Update data

To update an embedding, we only need to delete and add dimensions that have been modified. The AI models for each updated dimension are then retrained.

For a small number of data changes, we can also temporarily pause updating the AI models and only adjust the relevant search result processing code. Wait for more new data to be used to retrain AI models in each dimension.

D. Experiment

Comparing the FAISS IP search algorithm with our proposed Domain Search algorithm, the experimental results of using the same 100,000 English vocabulary on a Mac studio are as following table X:

TABLE X
COMPARISON OF FAISS IP AND DOMAIN SEARCH

	FAISS.IP	*Domain Search*	*Remark*
Mean time and standard deviation required to search for 1 embedding (30 samples)	0.0425, 0.0015	0.1721, 0.0323	A single dimension in Domain Search: 0.0358, 0.0107

As we can see from the table, Domain Search takes 4.05 times longer to search for an embedding than FAISS. However, domain search uses 5 AI models, and the search time of each AI model is 0.0358, which is less than the search time 0.0425 of FAISS. Therefore, if an AI model can search 5 dimensions at a time, the search speed of the optimized domain search could be lower than that of FAISS.

E. Discussion

Our embedding database can basically achieve the search complexity of O(5) on a single machine, and only a simple distributed search is needed to achieve the search complexity of O(1). When initializing the database, we need to train AI models for each dimension, and when the data is modified, we need to fine-tune the AI models. Therefore, this design is suitable for embedding databases with stable data or can be updated regularly. We can also use independent AI training servers to speed up database updates.

F. Next Step

The next step will be to explore ways to optimize Domain Search. The first is the multi-dimensional comprehensive search model, which searches for 5 dimensions in once, so that the O(1) search target can be completely realized. To achieve this, all we need to do is designing and training an AI model that can search 5 dimensions simultaneously with the same speed. The second is to simplify AI models to reduce the search time of a single AI model and the retraining time of AI models.

VI. CHAPTER 5, EMBEDDING COMPLETENESS

We believe that the actual dimension of a natural language can grow infinitely, but the model-based engineering dimension is limited. The quality of the embedding created within this finite engineering dimensions can be measured from the corpus on which it was trained. We continue to use the standard model in Chapter 3 to define the completeness of an embedding system: if the test loss in the full knowledge base training of a language system is less than epsilon, then the embedding system is considered complete. The experiment is to test those embeddings on any trained corpus using this standard model, and the test loss should be less than epsilon. In reality, there is no full knowledge base, so it is difficult for us to achieve global completeness. Let's add a criterion, which is local completeness. If an embedding system has all the knowledge of a local domain, then we say that the embedding system has local completeness. We should be able to test any corpus in this local domain with this standard model and get test loss less than epsilon.

A. Introduction

With the high-dimensional structure and proper training, an embedding system theoretically has the ability to learn any contextual knowledge and achieve a very high accuracy. Its completeness, i.e., its ability to remember all knowledge that it has learned, is doubly important. On the one hand, if embedding can't remember what has been learned, the AI model has the opportunity to hallucinate at these points. It can be considered that embedding with completeness is an important prerequisite for AI to eliminate hallucinations. On the other hand, if embedding cannot remember what it has learned, its downstream work, including all kinds of reasoning, can only lead to wrong conclusions. These errors can quickly infiltrate the entire neural network, polluting all branches downstream of the AI, thus hindering the evolution of AI to AGI.

The concept of completeness of embedding proposed by us refers to the idea that an embedding can contain all the knowledge embodied in the language system or the corpus by this word. A complete embedding system is a perfect substitute for a language system or a corpus and all the knowledge it contains, so that it can make perfect reasoning and successfully complete all the downstream tasks.

In this research, we are mainly exploring the establishment of a system for measuring the completeness of embedding.

B. Related Work

We found no research articles that addressed the completeness of embedding. The main reason is that researchers attribute the problem of hallucinations or the performance of AI to model architecture[66] or data problems[67]. But we think these issues are all very closely related to embedding. Most embedding assessment tools [35] are mainly aimed at the non-generative tasks, but we think that testing it by letting it retelling what it had learned before would be better.

C. Solution

Check if an embedding system is complete. It is to find a random corpus in the field where the embedding system claims to have completeness, use a GPT that only retains the embedding input layer and output logits layer to train up to 5000 epochs with patience = 500, and then do a sampling test, if the value of cross-entropy is less than epsilon, we consider it complete. Otherwise, it is not complete because it does not have full knowledge of the context in the sample documents, even though it has used them in training.

D. Experiments

We continue to use the embedding library generated by the Salesforce/SFR-Embedding-Mistral embedding model and the MSMARCO passage ranking database [68] that the researchers used to train this model to verify the completeness of this embedding system. We use the test model to directly input embedding, and output its generation results against the database, training 5000 steps, patience = 500, and finally get test loss = 3.1367, which cannot reach the threshold of 0.31. This experiment shows that the embedding system does not yet have local completeness, and the gap for the global completeness should be even greater.

E. Discussion

The test model proposed here is a framework proposal based on experimental experience, and the results are only used as a reference for further research. The more detailed setting of the test model requires a lot of experiments to confirm and theoretically proved. A complete embedding can lay the foundation for the establishment of the embedding standard library and prepare the way for the development of AGI.

F. Next Step

The next step is to study how to build a complete embedding system and try to first build an embedding system with local completeness, and then gradually expand the scope of the local area, so as to accumulate experience for the embedding with global completeness. Another important task is to verify and improve the completeness test model and its operating specifications.

VII. Chapter 6, Embedding Reasoning

Embedding reasoning is a central topic in AI. The breakthrough progress of large language models in reasoning in recent years has made the classic Turing test lose its significance as a benchmark for AI test. But there is still a huge confusion about how AI models acquire reasoning capabilities. This chapter will sort out all the existing forms of embedding reasoning in detail, propose some new reasoning algorithms, and make suggestions for the next evolution of AI reasoning architectures.

A. Introduction

AI reasoning is one or more embedding operations. From the perspective of vector operation, the basic operations of embedding are number operations. When natural language's basic unit is transformed from word to embedding, the operations of language are also changed from the symbolic operations of natural language to the number operations of embedding, and the picture of the world also changes from discrete to continuous, which is a fundamental transformation of great significance.

But as we can see from the discussion in Chapter 2, embedding, although it is represented as a group of very precise numbers, is still very broad in high-dimensional scope, a semantic cloud, and all vectors in this vast cloud have the same semantics. Thus, an embedding is just a representative of many embeddings in a semantic cloud. From this point of view, the operation of embedding is also the operation of the cloud of meaning, that is, the symbolic operation in the usual linguistic sense.

Considering that embedding is the complete embodiment of lexical semantics, embedding has the basis to be able to mine meaning and reason simply on its own. Embedding's inward digging and summarizing reasoning can be carried out on its own and does not require additional learning. But to deduce outward and discover new knowledge, it needs to learn rules beyond its semantics. Therefore, to judge whether embedding can rely on its own reasoning, it is only necessary to determine whether the knowledge introduced is implicit in or outside the semantics of the embedding word itself.

We divide embedding reasoning into three categories, the first is self-reasoning, that is, embedding does vector operations on itself. The second type is reasoning using simple tools, i.e., using an external vector or a kernel for operations. The third type is reasoning using complex tools, i.e., using external models consisting of multiple kernels for operations.

In the second part, we will review the relevant research on AI reasoning, then analyze the self-reasoning of embedding in the third part, analyze the reasoning of embedding using simple tools in the fourth part, analyze the reasoning of embedding using complex tools in the fifth part, conduct a comprehensive discussion in the sixth part, and finally look forward to the next steps in the seventh part.

B. Related Work

Embedding is a sequence of numbers, from a mathematical point of view, reasoning is a variety of operations, basic algebra, linear algebra, calculus, statistics, and probability theory are the main courses for humans to learn critical thinking. In the AI model, the various operations performed on embedding are the operations of linear algebra, calculus, statistics, and probability theory, so to speak, every step of the AI's operation is reasoning.

In order to help AI perform complex reasoning to achieve advanced intelligence, researchers have made various permutations and combinations of the above basic operations. The following table XI shows some famous representatives of AI reasoning algorithms that have been widely used.

TABLE XI
SOME FAMOUS EMBEDDING REASONING ALGORITHMS

Names	***Operations***	***Architecture***	***Related Papers***
Neural activation function	Basic algebra	Use alone	[69][70][71]
Feedforward algorithm	Linear algebra	Chain, network	[70] [72] [73]
Backpropagation	Calculus	Chain, network	[73] [72]
SoftMax	Statistics and Probability Theory	Network	[74]
Convolution	Linear algebra, Calculus	Parallel, series	[75]
Neural network	Linear algebra, Calculus	Series, network	[76] [70] [72]
LSTM	Linear algebra, Calculus	Series	[77]
Transformer	Linear algebra, Calculus	Parallel, series	[50]

Each algorithm has a clear reasoning purpose, such as Convolution to help AI discover the features of images, Backpropagation algorithm to help AI update knowledge, and Transformer to help AI remember the contextual meaning in a long text.

C. Embedding Self-Reasoning

As an embedding, it has memorized all the contextual circumstances in which it can appear. Therefore, to dig inward into the knowledge of its connotation is to open its semantic box. The main way to open the box is various embedding operations.

In addition to the operations we did in Chapter 2, multiplying a coefficient by a single or multiple dimensions, we can do some more operations.

1. Addition

Add one or more embeddings to get a new embedding. The semantics of the new embedding often fall into the original embedding domain or generate a new word with a similar meaning, as shown in Table XII.

TABLE XII
EMBEDDING ADDITION

Input Words	***Output Words (first choice, second choice)***
Daddy + mommy + son	dad, son
knife + fork + chopsticks	chopsticks, knife
man + woman	woman, man
sun + earth + mars	earth, sun
table + candle + chair	chair, table

Such results are more like sorting in a certain way.

2. Subtraction

Subtract one embedding from another to get a new embedding, as shown in Table XIII.

TABLE XIII
EMBEDDING SUBTRACTION

Input Words	***Output Words***
day - night	dayton
night - day	nighter
long - short	longworth
short - long	shorty
love - money	loveth
money - love	monetarily

3. Multiplication

Multiply one embedding elementwise by another embedding to get a new embedding. See Table XIV:

TABLE XIV
EMBEDDING MULTIPLICATION

Input Words	***Output Words***
love * money	revulsion
love * hate	scifi
love * war	fracas
artificial * intelligence	muerte
salt * food	triste

Most of the results of multiplication are negative words. This may reflect that the embedding system is not perfect, or that multiplication is not suitable for embedding operations, or that there is something else worth exploring.

As mentioned above, the operation of embedding is both continuous and symbolic. The discrete computation scope of natural language is huge, but the essence of embedding is infinite continuity. In order for humans to translate and understand infinitely continuous embedding in a huge domain of symbols, it is necessary to dig deeper into the characteristics of embedding, like setting a search distance threshold and making new word synthesis.

4. Division

By dividing one embedding elementwise by another embedding, we can get a new embedding, see Table XV.

TABLE XV
EMBEDDING DIVISION

Input Words	***Output Words***
freedom / democracy	joondalup
democracy / freedom	hypotenuse
home / money	metastases
money / home	kissimmee
worker / team	excempting
team / worker	pizzeria

After exploring the individual operations described above, we can also construct some combined operations that contain multiple basic operations.

These operations are the basic building blocks of massive computing in various complex AI architectures. Because we do not yet have a complete embedding system, this research is mainly used as an introduction, which needs to be further deepened. This is also one of the reasons why we currently have a macro understanding of AI, but its micro research is largely in a black box state.

5. global attention

For self-reasoning about an embedding sequence, we did the Moore-Penrose pseudo-inverse matrix and the PCA reconstruction matrix in Chapter 2. Here we propose another algorithm, Global Attention, which is mainly used to discover the topic word of an embedding sequence. Our Global Attention algorithm borrows from the classic self-attention algorithm [50], but remove the trichotomy of Q, K, and V, and uniformly use its original embedding. Because there are no additional training steps, it's actually a synthesis of the element-wise multiplication algorithm. After the first multiplication, use SoftMax to calculate the weights, and then update each embedding. This is done several times until all embeddings are converted into the same embedding. This embedding is the central word of this sequence of embeddings. Here are some examples in the table XVI.

TABLE XVI
GLOBAL ATTENTION OPERATION

Examples	***Global Attention Discovered Central Word***
'do not teach fish to swim'	teach
'an apple a day keeps the doctor away'	doctor
'a surface is that which has length and breadth only'	length
'the last thing you figure out in writing a book is what to put first'	writing
'Wikipedia is a free online encyclopedia, created and edited by volunteers around the world and hosted by the Wikimedia Foundation.'	world

D. Embedding Reasoning with Simple Tools

Embedding reasoning with simple tools mainly refers to the use of universal reasoning tools to help embedding to reason. In the case of an embedding or a sequence of embeddings, a simple tool is usually a kernel.

For example, in the syllogism [78] , for categorical syllogism, we have a major premise of length 4 and a minor premise of length 4, and we only need to use a kernel such as $formula(13)$.

$$\begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{13}$$

Let's use the major premise and minor premise to form a $2 \times 4 \times 4096$ tensor, and then multiply this tensor with this kernel to get a correct syllogism.

In the same way for hypothetical syllogism, we can have a corresponding kernel algorithm (14) to deal with:

$$\begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \tag{14}$$

For all basic reasoning, we can convert the logical operations into kernel operations and apply them to embeddings, simplifying and standardizing the embedding reasoning process.

E. Embedding Reasoning with Complex Tools

Reasoning that requires a lot of knowledge beyond semantics is a complex reasoning process, which is mainly undertaken by our various large and small language models, such as ChatGPT [79], Llama [80], Claude [81], Gemini [82], Grok [21], Mistral [83], etc.

Complex AI models are able to perform complex reasoning mainly because they use thousands of trainable kernels, that is, the weights of each unit at each layer. The learning ability of large language models is mainly reflected in the massive number of learnable kernels, and the knowledge learned by them is also stored in these kernels. Therefore, theoretically, the more parameters of the model, the more and larger the kernel, the stronger its learning and reasoning ability, and the more knowledge it can store. This is the intrinsic microscopic reason why the Scaling Laws work. All in all, A large amount of semantic knowledge and powerful reasoning capabilities are stored in high-dimensional embedding and these kernels, giving AI models amazing achievements.

The power of complex tools requires the same order of magnitude of data and high-dimensional embedding to match the amount of data. In a large language model that integrates the embedding model and the reasoning model, its input and output are either embedding or some transformation form of embedding, and the result of each kernel operation between layers or between operation units can be regarded as an intermediate of final embedding. Users of AI models do not need to worry about the existence of these intermediates, but researchers can extract and detect these intermediates to diagnose the operations of AI. While no one has published detailed vivisection of a large AI model, this work is key to our understanding of AI. The core point is to understand the meaning of embedding of these intermediates and the algorithms and functions of each internal kernel. Despite the huge size of large language models, most of them are repetitive structures, so it is not as incredible to dissect an AI model as people might think. We only need to study the functions and inputs and outputs of some key modules to have a good grasp of an AI model. However, there is still a big difference between vivisection and static analysis [84], because static analysis is difficult to detect and analyze the emergence of intelligence, which is the core secret of the development of AI.

F. Discussion

One of the main reasons why AI often makes some low-level mistakes when using complex models for high-level reasoning is that the reasoning operations in large AI models are basically disconnected from the basic reasoning operations of embedding and the intermediate reasoning with simple tools. The design of advanced reasoning algorithms in AI models does not consciously use embedding basic reasoning and intermediate reasoning algorithms. The basic logic operations that should be completely solved in the intermediate reasoning algorithms have not become an integral part of the current AI advanced reasoning algorithms. Can AI learn logical principles based solely on the context of text? It seems that this is a question mark to be verified. In the design of AI models, how to consciously and systematically integrate various algorithms of embedding primary and intermediate reasoning into the complex architecture of AI is an important direction for the next evolution of AI.

The evolution process from machine learning to AI is an intelligent explosion brought by the mutation and evolution of the basic elements of embedding and the basic computing unit of transformer. If the world is really spiraling, we can expect an upgraded version of machine learning period, that is, machine learning based on embedding as basic element and transformer as basic computing unit, this machine learning upgrade process can simplify the AI reasoning process and lower learning and use costs, but also conducive to the universal penetration and low-cost application of AI, and truly bring profound changes to all aspects of human life.

G. Next Step

The next step in research can be carried out in many ways. The first is to extensively create primary, intermediate level embedding reasoning algorithms, for which a basic algorithm library can be established. The second is to innovate the AI architecture design, which comprehensively integrates primary and intermediate reasoning algorithms in AI advanced reasoning algorithms, which could greatly reduce AI hallucinations. The third is to deeply dissect a large language model, analyze the functions and inputs and outputs of each internal kernel in detail, and analyze the meaning change flow of each embedding intermediate. Fourth, the traditional machine learning field is upgraded with the in-depth use of new embedding basic components and transformer basic computing units, so that AI could penetrate into all corners of human life, learning, work and scientific research at a low cost, and comprehensively improve the intelligence level and operation quality of human society.

VIII. Chapter 7, General Embedding

The 'cat' in human intelligence has not only the word 'cat' but also the sound and image of the cat. In contrast, the embedding of 'cat' in AI language only contains the textual semantics of 'cat'. Therefore, we also need to transfer the sound and image of the cat to the world of AI through embedding, to help AI understand the real world more deeply through embedding. Just as the human brain can understand the colorful real world through a unified format of neural signals from various senses, AI can also in the same way to understand the real world by a unified form of embedding.

A. Introduction

When the AI thinks about the world, all the physical signals in the world need to be transformed into embedding. A person's perception of the weight of 1 pound and the feeling of a length of 1 foot both come from a reference to the person own strength and height. AI doesn't have a physical presence; how do they get that feeling? It can be seen here that AI needs absolutely objective cognition, not personalized feelings. Everyone has a different feeling about the weight of 1 pound, but what AI needs is a consistent and absolute perception. Therefore, the cognitive perspective of AI is an omnipotent perspective.

The recognition of the omnipotent perspective of the physical world, detached from the personal perspective, is exactly what our scientific research pursues. Therefore, AI can obtain an objective understanding of the physical world by reading scientific literature, without the need to obtain sensory signals with the help of its own body, as humans do. Just like cosmologists don't need to physically go to a star to study it, AI doesn't need to actually 'go' to our real world to understand it.

Read more of the literature is actually a very reliable learning method for AI, including reading images, sounds, and videos, of course. Multimodal learning can help AI better understand the world, and it is important that all embedding, including text, images, sounds, videos, and all other embedding systems, remain internally consistent. In other words, the embedding of the word 'cat' is consistent with the embedding of images, sounds, videos and various physical signals about cats. Therefore, the process of embedding the word 'cat' is the process of embedding more text, images, sounds, videos and various related physical signals, so that the embedding of pure text of 'cat' evolves into a general embedding containing multimodal information. After all the words have been perfected converted into general embeddings, the basic language of the AI embedding is ready to evolve into the AGI language.

In the next part, we will review some of the important research work, then propose our new solution in part 3, conduct an innovative experiment in part 4, discuss it further in part 5, and propose next steps in part 6.

B. Related Work

Before the advent of multimodal models, fusion of embedding was mainly the merging of different text embeddings, generally directly or weighted to do addition. Some are element-wise addition[55], and embedding concatenation is also common[85]. In fact, these two additions have thousands of applications in all AI models and are the main way for large language models to achieve internal fusion. After the emergence of multimodal models, it is necessary to fuse the embedding of different data. Since the dimensions of audio, video, and image embedding may be different, and element-wise addition cannot be done directly, the common method is concatenation. Facebook has[86] a prime example. Let's take a look at its model architecture in Figure 4:

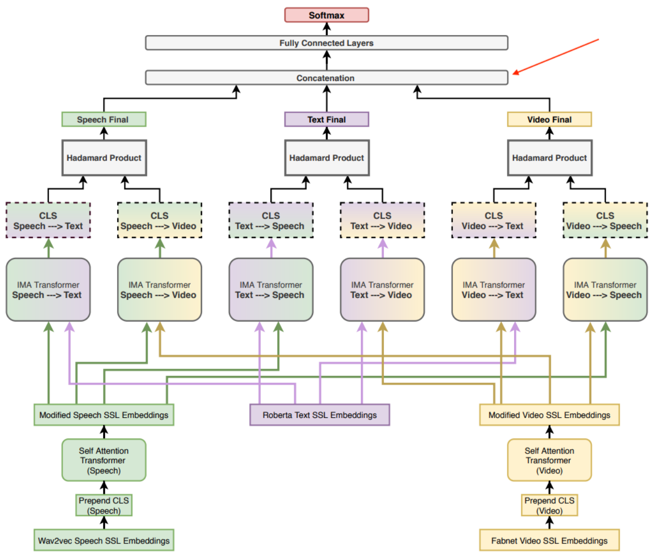
Softmax
Fully Connected Layers
Concatenation
Speech Final
Text Final
Video Final
Hadamard Product
Hadamard Product
Hadamard Product
CLS Speech ---> Text
CLS Speech ---> Video
CLS Text ---> Speech
CLS Text ---> Video
CLS Video ---> Text
CLS Video ---> Speech
IMA Transformer Speech ---> Text
IMA Transformer Speech ---> Video
IMA Transformer Text ---> Speech
IMA Transformer Text ---> Video
IMA Transformer Video ---> Text
IMA Transformer Video ---> Speech
Modified Speech SSL Embeddings
Roberta Text SSL Embeddings
Modified Video SSL Embeddings
Self Attention Transformer (Speech)
Self Attention Transformer (Video)
Prepend CLS (Speech)
Prepend CLS (Video)
Wav2vec Speech SSL Embeddings
Fabnet Video SSL Embeddings

Fig. 4. Overview of the Self Supervised Embedding Fusion Transformer (SSE-FT), from 'Multimodal Emotion Recognition With Transformer-Based Self Supervised Feature Fusion'

As we can see from the architecture diagram of Facebook AI model, after the final layer of the multimodality, there is a fusion layer, which uses concatenation, and then uses a fully connected layer to adjust the dimension, and then hands it over to the SoftMax layer to calculate the probability. The benefit of this architecture design is that it is consistent with the interface of the existing AI model architecture. But the problem is that its embedding must be completed at one time, and when updating the embedding, all the knowledge of each modality must be relearned, and it is difficult to expand the knowledge in increment manner. In the following sections, we provide an overview of this traditional fusion approach and then introduce our innovative Embedding Augmenting approach.

C. Solution

1. Fusion

Before training general embedding, in addition to a large text library, we also need to collect and organize a large video library, sound library, image library, and other related physical signal libraries. Then, when training general embedding, use all of these libraries, and the trained embedding evolves from traditional text embedding to general embedding.

There are also two methods of general embedding upgrade: database upgrade and model upgrade. Among them, the upgrade of the database includes the upgrading of various text libraries, image libraries, audio libraries, video libraries and various physical signal libraries, then retrain the model using the updated libraries. Sometimes, we need to redesign the model, and train the new model on the libraries, this is a model upgrade only.

One challenge to note here is choosing a suitable addition strategy. If we are using element-wise addition, we need to use a uniform embedding setting to embed knowledge in various libraries. When we use 0-255 embedding for an image, we need to convert this embedding to the same dimension of decimal format as word embedding. In the same way, after we quantize a piece of sound, we also need to convert it to embedding in the same dimension as word embedding, and then we can merge them with word embedding. If we use sequential addition, i.e., concatenation, we generally need to normalize the values of various embeddings to a similar range of values. For example, if we use the value range of word embedding in Chapter 2 from -120.926445 to 137.37656, then the value of our audio, video and image embedding should also be in the same range, so that the fusion effect is better.

2. Embedding Augmenting

According to the findings of Chapter 2, embedding has many redundant dimensions. Our proposed Embedding Augmenting solution takes advantage of these redundant dimensions to change the value of some dimensions to the embedding value of the image or audio that needs to be grafted, without changing the semantics of embedding. This is similar to gene grafting technology, which replaces some old genes with new genes without changing the species. In this way, the new embedding still has the original semantics but already carries some new image or sound information, which is a richer and augmented general embedding.

D. Experiment

Meta's paper and other papers[87][88]have discussed the multimodal fusion approach in depth, mainly using various addition operations, and will not be tested here. We mainly conducted experiment of Embedding Augmenting.

First, we use OpenCV [89] to convert an image of a cat to grayscale and resize it to 30x30 pixels to get 900 image data. In the last 900 dimensions of the 4096-dimensional word embedding of the 'cat', we replaced the original values of these dimensions with these pixel grayscale values. We search for this new embedding with FAISS and get the same semantics as 'cat'. But the image of the cat has been recorded into the embedding of the 'cat'. When an AI with higher intelligence reads this embedding, it is able to recognize the image of the cat.

Fig. 5. A cat, Photo by Alvan Nee on Unsplash

We then used $torchaudio$[90] to sample a cat meow record at 4000Hz for 0.1 seconds to obtain 400 data, and embedded this audio data into the first 400 dimensions of the embedding of the 'cat's embedded image information to replace the values of the original dimensions. FAISS searches for this new embedding with image and sound information, and the semantics obtained is still 'cat'. When an AI with image and sound intelligence reads this new embedding, it can not only understand that the semantics of this embedding is 'cat', but also see the image of the cat and hear the cat's meow simultaneously.

Embedding augmenting is an incremental technology, which sets functional blocks in the dimension space of embedding. This approach can be based on the original embedding with the expansion of understanding. Only the corresponding dimensions in the original embedding need to be gradually revised without having to tear down all the dimensional values and retrain. It's like human learning process, we can learn about the text of the cat first, then learn about the cat image, then get familiar with the cat's meow, and other things about the cat. As knowledge continues to increase, so does the understanding of cat. In this process of accumulating knowledge, we do not need to change the way the text is spelled, but the knowledge about cat has increased. With embedding augmenting technology, AI can do the same.

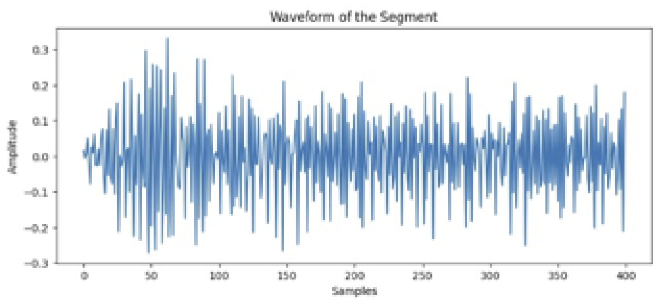

Fig. 6. Cat meow audio sampling data

E. Discussion

With the standardization of embedding, we can hopefully put all the knowledge about cat into one embedding of 'cat'. This will be of great help to the improvement of the level of AI.

F. Next Step

There are a lot of next steps on General Embedding. The first is to carry out standardized exploration of embedding, such as dimension setting and functional division. The second is to design an advanced embedding training model, which can train text embedding and be able to add and modify the embedding dimension of images, sounds, or other information at any time. The third is to design a high-level AI model that can fully understand all kinds of embedding information, including semantics, images, sounds, and other information.

IX. CHAPTER 8, EMBEDDING EVERYTHING

Embedding everything is to reformat everything in the real world into continuous embeddings in high-dimensional space according to the settings of AI language, to facilitate the understanding and analysis of AI. Embedding everything is also the basis for AI to be able to deeply perceive and understand the world and evolve into AGI. From a linguistic point of view, Embedding's technique is mainly used to mine the meaning of symbols from a high-dimensional space. This symbol does not have to be a symbol in the language, but of course other symbol systems are also applicable. For example, animal language, physical signals, chemical signals, as well as the image information and audio signals we studied in the previous chapter, can all be well suited to embedding techniques. The information to be mined is not limited to being discrete, continuous information can also be transformed into AI language in high-dimensional space through embedding. This chapter takes the widely used transaction price information of the economic market as an example to explore the application of embedding technology in the field of continuous information.

A. Introduction

The process of embedding is generally a learning process in AI. Because the price information is continuous, we can directly use a weight kernel or a group of weight kernels that can learn to map the price information into a high-dimensional space and convert a price series into an embedding sequence. In this way, AI can easily analyze this price information from a high-dimensional space, dig into the deep interrelationships, and provide better predictions for future prices or price movements.

B. Related Work

Market transaction information generally refers to the transaction price, trading volume, and other relevant transaction information, the most important is the transaction price. The trading price is a one-dimensional, continuous variable, and theoretically it has an infinite number of possible values. Therefore, it is not like a symbol in the usual sense, with only a finite number of possible values. To dig deeper into the meaning of a one-dimensional continuous variable in a high-dimensional space, we only need to map it directly into a high-dimensional space and create its embedding directly, without first building a vocabulary like natural language with a symbol system. In addition to using language models to predict transaction sentiment by using text processing of relevant news and posts [91][92], researchers also use transformer architecture to process transaction price information and predict its future changes[92][93]. The relevant part of the general workflow is to first map the one-dimensional transaction price information into multiple dimensions of embedding through one or more fully connected neural network layers, and then process it by the transformer architecture of the multi-head attention mechanism.

Of course, if we want to simplify the price information, reduce the noise, especially reduce the impact of some extreme trading prices on the forecast, we can also consider building a trading price vocabulary according to the precision requirements, that is, establishing a trading language system, and then translating the original trading price data into trading language sentences using the prescribed trading vocabulary. These sentences are then converted into embedding sequences.

There are reasons to believe that every word in a trading sentence is influenced by a number of words that precede it. That is, the current trading price is affected by the previously traded prices. It's common knowledge that words in English sentences are influenced by words that precede them. All sentences in a trading language for a long period of time, such as several years, is a trading library, and an embedding system can be trained on it. This embedding system can help the AI model predict the next token, that is, the next transaction price or its movement, based on the previous transaction sentence, to help make better trading decisions.

It should be noted that there is a subtle difference between the results obtained by the two embedding methods. The embedding created from the price vocabulary has a space volume, because its original data has a space length. However, if a fully connected neural network layer is used to directly convert the point of a continuous variable into an embedding in a high-dimensional space, the original data and the final embedding are only a point in the space, and theoretically this embedding does not seem to have a domain volume. Can this primitive price point gain volume in the process of explosively expanding space from 1 to 4096 dimensions? This can be done with reference to the Big Bang theory about the origin of the universe [94][95] or a specialized experiment can be designed to test it in a high-dimensional transactional linguistic space.

C. Experiment

We used the hourly transaction data of BTC from 2015 to 2021, trained an AI decision-making model according to the above method, then tested it with the actual data in 2022, and finally achieved an annual return of 10.32%, the Buy & Hold strategy gains was -59.22%. Then we added the market transaction data of 2022 to the training dataset, retrained the model, and tested it with the real data from 2023, finally achieved an annual return of 305.63%, compared to the market return of 257.12% in 2023. Our experiments show that embedding-based AI trading decision-making models have great potential.

D. Discussion

Different commodities and securities seem to be traded in different price languages, but they can all be translated into the same language in terms of the rate of change in trading prices. From the perspective of embedding, they can also be in the same high-dimensional space. If all the trading market data [96] translated into the same trading language, it will become a quasi 'complete knowledge base' of the trading market, and will the AI trained with this huge library surprise us like a large language model? This is subject to a lot of experimentation and deeper exploration.

Is the next token in a transaction sentence only affected by a few tokens before it? Is the next word in an English sentence only affected by a few words before it? These two questions ask the same question. A deeper exploration of the intrinsic nature of trading sentences is an inevitable requirement for the application of embedding techniques in the trading market. The key here is the amount of data, compared to the huge amount of data currently used to train large language models, perhaps training a large market language model of the trading market with a giant database of various trading market data collected over 300 years can really reveal the answer to the above questions and solve the mystery of market efficiency.

E. Next Step

Whether it is continuous data such as transaction prices, or discrete symbolic sequences such as natural languages, it can be well transformed into embeddings and become input data for AI models. Theoretically, this provides a technical possibility for AI to analyze everything in the world in any high-dimensional space, paves the way for AI to widely penetrate into all aspects of human world, and of course opens the door for AI to evolve into AGI and ASI.

X. Postscript

Human intelligence is low-dimensional and discrete, while artificial intelligence is high-dimensional and continuous.

Our research starts with the fundamental properties of embedding as an AI language, delves into the unique properties of embedding, and explores the dimension setting of embedding, the basic architecture design principles of AI model based on these characteristics, the new O(1) search algorithm and the design of embedding database, the completeness of embedding and embedding reasoning, and the idea of transforming the entire real world into embedding. We have mainly established some basic research frameworks for embedding as AI language, paving the way for broader and more in-depth AI linguistics research.

AI will be destined to become a higher level of intelligence beyond human intelligence, the fundamental reason is that it can manipulate higher dimensions, and there is no theoretical upper limit to the intelligent dimensions of AI. It is expected that a new round of science and technology explosion will be born from AGI and ASI. In-depth understanding of AI language embedding is not only about mutual understanding and smooth communication between humans and AI, but also about the development and future destiny of human beings.

References

[1] V. Di Maio and S. Santillo, "Information Processing and Synaptic Transmission," in Advances in Neural Signal Processing, R. Vinjamuri, Ed., IntechOpen, 2020. doi: 10.5772/intechopen.88405.

[2] J. L. S. Bellmund, P. Gärdenfors, E. I. Moser, and C. F. Doeller, "Navigating cognition: Spatial codes for human thinking," Science, vol. 362, no. 6415, p. eaat6766, Nov. 2018, doi: 10.1126/science.aat6766.

[3] T. Diyora, "VOCABULARY AS AN ADAPTIVE SYSTEM AND UNDERSTANDING LANGUAGE DEVELOPMENT," Education News Research In The 21st Century, vol. 2, no. 19, Art. no. 19, Mar. 2024.

[4] M. Zhang, O. Press, W. Merrill, A. Liu, and N. A. Smith, "How Language Model Hallucinations Can Snowball," May 22, 2023, arXiv: arXiv:2305.13534. doi: 10.48550/arXiv.2305.13534.

[5] J. V. Atanasoff et al., "The Invention of Electronic Digital Computing - Plenary Panel Summary," in 2023 IEEE John Vincent Atanasoff International Symposium on Modern Computing (JVA), Chicago, IL, USA: IEEE, Jul. 2023, pp. 8–8. doi: 10.1109/JVA60410.2023.00011.

[6] J. Singh and M. Singh, "Evolution in Quantum Computing," in 2016 International Conference System Modeling & Advancement in Research Trends (SMART), Moradabad, India: IEEE, 2016, pp. 267–270. doi: 10.1109/SYSMART.2016.7894533.

[7] H. R. Lewis, Ed., Ideas That Created the Future: Classic Papers of Computer Science. The MIT Press, 2021. doi: 10.7551/mitpress/12274.001.0001.

[8] S. Robertson, "Understanding inverse document frequency: on theoretical arguments for IDF," J. Doc., vol. 60, no. 5, pp. 503–520, Jan. 2004, doi: 10.1108/00220410410560582.

[9] E. Arnaud, M. Elbattah, M. Gignon, and G. Dequen, "NLP-Based Prediction of Medical Specialties at Hospital Admission Using Triage Notes," in 2021 IEEE 9th International Conference on Healthcare Informatics (ICHI), Victoria, BC, Canada: IEEE, Aug. 2021, pp. 548–553. doi: 10.1109/ICHI52183.2021.00103.

[10] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," Sep. 06, 2013, arXiv: arXiv:1301.3781. Accessed: Mar. 22, 2024. [Online]. Available: http://arxiv.org/abs/1301.3781

[11] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," Oct. 16, 2013, arXiv: arXiv:1310.4546. Accessed: Mar. 22, 2024. [Online]. Available: http://arxiv.org/abs/1310.4546

[12] L. Rheault and C. Cochrane, "Word Embeddings for the Analysis of Ideological Placement in Parliamentary Corpora," Polit. Anal., vol. 28, no. 1, pp. 112–133, Jan. 2020, doi: 10.1017/pan.2019.26.

[13] S. Jansen, "Word and Phrase Translation with word2vec," Apr. 24, 2018, arXiv: arXiv:1705.03127. Accessed: Mar. 22, 2024. [Online]. Available: http://arxiv.org/abs/1705.03127

[14] Q. V. Le and T. Mikolov, "Distributed Representations of Sentences and Documents," May 22, 2014, arXiv: arXiv:1405.4053. Accessed: Mar. 22, 2024. [Online]. Available: http://arxiv.org/abs/1405.4053

[15] I. Banerjee, M. C. Chen, M. P. Lungren, and D. L. Rubin, "Radiology report annotation using intelligent word embeddings: Applied to multi-institutional chest CT cohort," J. Biomed. Inform., vol. 77, pp. 11–20, Jan. 2018, doi: 10.1016/j.jbi.2017.11.012.

[16] "MTEB Leaderboard - a Hugging Face Space by mteb." Accessed: Mar. 23, 2024. [Online]. Available: https://huggingface.co/spaces/mteb/leaderboard

[17] "SFR-Embedding-Mistral: Enhance Text Retrieval with Transfer Learning," Salesforce AI. Accessed: Mar. 23, 2024. [Online]. Available: https://blog.salesforceairesearch.com/sfr-embedded-mistral/

[18] Y. Wang, Y. Hou, W. Che, and T. Liu, "From static to dynamic word representations: a survey," Int. J. Mach. Learn. Cybern., vol. 11, no. 7, pp. 1611–1630, Jul. 2020, doi: 10.1007/s13042-020-01069-8.

[19] J. Von Der Mosel, A. Trautsch, and S. Herbold, "On the Validity of Pre-Trained Transformers for Natural Language Processing in the Software Engineering Domain," IEEE Trans. Softw. Eng., vol. 49, no. 4, pp. 1487–1507, Apr. 2023, doi: 10.1109/TSE.2022.3178469.

[20] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations," Feb. 08, 2020, arXiv: arXiv:1909.11942. Accessed: Mar. 23, 2024. [Online]. Available: http://arxiv.org/abs/1909.11942

[21] xai-org/grok-1. (Mar. 23, 2024). Python. xai-org. Accessed: Mar. 23, 2024. [Online]. Available: https://github.com/xai-org/grok-1

[22] M. Schreiner, "GPT-4 architecture, datasets, costs and more leaked," THE DECODER. Accessed: Mar. 23, 2024. [Online]. Available: https://the-decoder.com/gpt-4-architecture-datasets-costs-and-more-leaked/

[23] S. Jindal, "Meta To Release Llama 3 in July, Outperforms GPT-4 and Gemini," Analytics India Magazine. Accessed: Mar. 23, 2024. [Online]. Available: https://analyticsindiamag.com/meta-to-release-llama-3-in-july-outperforms-gpt-4-and-gemini/

[24] D. A. D. Thompson, "The Memo - Special edition: Claude 3 Opus," The Memo by LifeArchitect.ai. Accessed: Mar. 23, 2024. [Online]. Available: https://lifearchitect.substack.com/p/the-memo-special-edition-claude-3

[25] "Embeddings," Voyage AI. Accessed: Mar. 23, 2024. [Online]. Available: https://docs.voyageai.com/docs/embeddings

[26] kalyan, "GPT-4 Everything You Need To Know," MAKE ME ANALYST. Accessed: Mar. 23, 2024. [Online]. Available: https://makemeanalyst.com/gpt-4-everything-you-need-to-know/

[27] J. Kaplan et al., "Scaling Laws for Neural Language Models," Jan. 22, 2020, arXiv: arXiv:2001.08361. doi: 10.48550/arXiv.2001.08361.

[28] S. K. Routray, A. Javali, K. P. Sharmila, M. K. Jha, M. Pappa, and M. Singh, "Large Language Models (LLMs): Hypes and Realities," in 2023 International Conference on Computer Science and Emerging Technologies (CSET), Oct. 2023, pp. 1–6. doi: 10.1109/CSET58993.2023.10346621.

[29] M. T. Bennett, "Enactivism & Objectively Optimal Super-Intelligence," Mar. 08, 2023, arXiv: arXiv:2302.00843. Accessed: Mar. 23, 2024. [Online]. Available: http://arxiv.org/abs/2302.00843

[30] "How many words are there in English? — Merriam-Webster." Accessed: Jul. 23, 2024. [Online]. Available: https://www.merriam-webster.com/help/faq-how-many-english-words

[31] "An English Word List." Accessed: Jul. 23, 2024. [Online]. Available: https://websites.umich.edu/ jlawler/wordlist.html

[32] "Wikimedia Downloads." Accessed: May 28, 2024. [Online]. Available: https://dumps.wikimedia.org/

[33] facebookresearch/faiss. (Jun. 10, 2024). C++. Meta Research. Accessed: Jun. 09, 2024. [Online]. Available: https://github.com/facebookresearch/faiss

[34] J. Chen and G. de Melo, "Semantic Information Extraction for Improved Word Embeddings," in Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing, P. Blunsom, S. Cohen, P. Dhillon, and P. Liang, Eds., Denver, Colorado: Association for Computational Linguistics, Jun. 2015, pp. 168–175. doi: 10.3115/v1/W15-1523.

[35] N. Muennighoff, N. Tazi, L. Magne, and N. Reimers, “MTEB: Massive Text Embedding Benchmark,” Mar. 19, 2023, arXiv: arXiv:2210.07316. Accessed: Jan. 28, 2024. [Online]. Available: http://arxiv.org/abs/2210.07316

[36] P. Mani et al., “The Hubness Phenomenon in High-Dimensional Spaces,” in Research in Data Science, E. Gasparovic and C. Domeniconi, Eds., Cham: Springer International Publishing, 2019, pp. 15–45. doi: 10.1007/978-3-030-11566-1_2.

[37] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”.

[38] E. W. Weisstein, “Moore-Penrose Matrix Inverse.” Accessed: Jul. 23, 2024. [Online]. Available: https://mathworld.wolfram.com/

[39] E. Sezerer and S. Tekir, “A Survey On Neural Word Embeddings,” Oct. 04, 2021, arXiv: arXiv:2110.01804. doi: 10.48550/arXiv.2110.01804.

[40] S. Liu, C. Gao, Y. Chen, D. Jin, and Y. Li, “Learnable Embedding Sizes for Recommender Systems,” Mar. 11, 2021, arXiv: arXiv:2101.07577. doi: 10.48550/arXiv.2101.07577.

[41] B. He, X. He, R. Zhang, Y. Zhang, R. Tang, and C. Ma, “Dynamic Embedding Size Search with Minimum Regret for Streaming Recommender System,” in Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, Oct. 2023, pp. 741–750. doi: 10.1145/3583780.3615135.

[42] X. Zhao et al., “AutoDim: Field-aware Embedding Dimension Searchin Recommender Systems,” in Proceedings of the Web Conference 2021, Ljubljana Slovenia: ACM, Apr. 2021, pp. 3015–3022. doi: 10.1145/3442381.3450124.

[43] Y. Wang, “Single Training Dimension Selection for Word Embedding with PCA,” in Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), K. Inui, J. Jiang, V. Ng, and X. Wan, Eds., Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 3597–3602. doi: 10.18653/v1/D19-1369.

[44] Z. He, M. Pan, Y. Wang, G. Xu, W. Su, and C. Shi, “The Influence of Embedding Size and Hidden Units Parameters Changes on the Sequence2Sequence Model,” in 2022 5th International Conference on Artificial Intelligence and Big Data (ICAIBD), May 2022, pp. 150–155. doi: 10.1109/ICAIBD55127.2022.9820494.

[45] “New and improved embedding model.” Accessed: May 28, 2024. [Online]. Available: https://openai.com/index/new-and-improved-embedding-model/

[46] “The Llama 3 Herd of Models — Research - AI at Meta.” Accessed: Jul. 27, 2024. [Online]. Available: https://ai.meta.com/research/publications/the-llama-3-herd-of-models/

[47] D. Archer, “Re-Recognize Overfitting in Neural Network.”

[48] “English-Corpora: COCA.” Accessed: May 28, 2024. [Online]. Available: https://www.english-corpora.org//coca/

[49] “American Stories.” Accessed: May 28, 2024. [Online]. Available: https://dell-research-harvard.github.io/resources/americanstories

[50] A. Vaswani et al., “Attention Is All You Need,” Aug. 01, 2023, arXiv: arXiv:1706.03762. doi: 10.48550/arXiv.1706.03762.

[51] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, “Improving Language Understanding by Generative Pre-Training”.

[52] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language Models are Unsupervised Multitask Learners”.

[53] D. Hendrycks and K. Gimpel, “Gaussian Error Linear Units (GELUs),” Jun. 05, 2023, arXiv: arXiv:1606.08415. doi: 10.48550/arXiv.1606.08415.

[54] J. L. Ba, J. R. Kiros, and G. E. Hinton, “Layer Normalization,” Jul. 21, 2016, arXiv: arXiv:1607.06450. doi: 10.48550/arXiv.1607.06450.

[55] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” Dec. 10, 2015, arXiv: arXiv:1512.03385. doi: 10.48550/arXiv.1512.03385.

[56] Y. Han, C. Liu, and P. Wang, “A Comprehensive Survey on Vector Database: Storage and Retrieval Technique, Challenge,” Oct. 18, 2023, arXiv: arXiv:2310.11703. doi: 10.48550/arXiv.2310.11703.

[57] J. L. Bentley, “Multidimensional binary search trees used for associative searching,” Commun. ACM, vol. 18, no. 9, pp. 509–517, Sep. 1975, doi: 10.1145/361002.361007.

[58] M. Dolatshah, A. Hadian, and B. Minaei-Bidgoli, “Ball*-tree: Efficient spatial indexing for constrained nearest-neighbor search in metric spaces,” Nov. 02, 2015, arXiv: arXiv:1511.00628. doi: 10.48550/arXiv.1511.00628.

[59] A. Guttman, “R-trees: a dynamic index structure for spatial searching,” ACM SIGMOD Rec., vol. 14, no. 2, pp. 47–57, Jun. 1984, doi: 10.1145/971697.602266.

[60] “M-Tree: An Efficient AccessMethod for Similarity Search in Metric Spaces”.

[61] O. Jafari, P. Maurya, P. Nagarkar, K. M. Islam, and C. Crushev, “A Survey on Locality Sensitive Hashing Algorithms and their Applications,” Feb. 17, 2021, arXiv: arXiv:2102.08942. doi: 10.48550/arXiv.2102.08942.

[62] X. Luo et al., “A Survey on Deep Hashing Methods,” Apr. 23, 2022, arXiv: arXiv:2003.03369. doi: 10.48550/arXiv.2003.03369.

[63] “Approximate Nearest Neighbors Oh Yeah (ANNOY) - AAU Social Data Science Deep Learning - 2019 Portfolio.” Accessed: Jun. 09, 2024. [Online]. Available: https://sds-aau.github.io/M3Port19/portfolio/ann/

[64] Y. A. Malkov and D. A. Yashunin, “Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs,” Aug. 14, 2018, arXiv: arXiv:1603.09320. doi: 10.48550/arXiv.1603.09320.

[65] T. Ge, K. He, Q. Ke, and J. Sun, “Optimized Product Quantization for Approximate Nearest Neighbor Search,” in 2013 IEEE Conference on Computer Vision and Pattern Recognition, Jun. 2013, pp. 2946–2953. doi: 10.1109/CVPR.2013.379.

[66] Y. Zhang et al., “Siren’s Song in the AI Ocean: A Survey on Hallucination in Large Language Models,” Sep. 24, 2023, arXiv: arXiv:2309.01219. doi: 10.48550/arXiv.2309.01219.

[67] “When AI Gets It Wrong: Addressing AI Hallucinations and Bias,” MIT Sloan Teaching & Learning Technologies. Accessed: Jul. 23, 2024. [Online]. Available: https://mitsloanedtech.mit.edu/ai/basics/addressing-ai-hallucinations-and-bias/

[68] “MSMARCO-Passage-Ranking,” MSMARCO-Passage-Ranking. Accessed: Jul. 24, 2024. [Online]. Available: https://microsoft.github.io/MSMARCO-Passage-Ranking/

[69] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” Bull. Math. Biophys., vol. 5, no. 4, pp. 115–133, Dec. 1943, doi: 10.1007/BF02478259.

[70] F. Rosenblatt, “The perceptron: A probabilistic model for information storage and organization in the brain,” Psychol. Rev., vol. 65, no. 6, pp. 386–408, 1958, doi: 10.1037/h0042519.

[71] M. A. Nielsen, Neural Networks and Deep Learning. Determination Press, 2015.

[72] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” Nature, vol. 323, no. 6088, pp. 533–536, Oct. 1986, doi: 10.1038/323533a0.

[73] P. J. Werbos, The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting. John Wiley & Sons, 1994.

[74] M. Costa, “Probabilistic interpretation of feedforward network outputs, with relationships to statistical prediction of ordinal quantities,” Int. J. Neural Syst., vol. 7, no. 5, pp. 627–637, Nov. 1996, doi: 10.1142/s0129065796000610.

[75] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proc. IEEE, vol. 86, no. 11, pp. 2278–2324, Nov. 1998, doi: 10.1109/5.726791.

[76] B. Widrow and M. E. Hoff, “(1960) Bernard Widrow and Marcian E. Hoff, ‘Adaptive switching circuits,’ 1960 IRE WESCON Convention Record, New York: IRE, pp. 96-104,” in Neurocomputing, Volume 1, J. A. Anderson and E. Rosenfeld, Eds., The MIT Press, 1988, pp. 126–134. doi: 10.7551/mitpress/4943.003.0012.

[77] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” Neural Comput., vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[78] P. Shorey, “The Origin of the Syllogism,” Class. Philol., vol. 19, no. 1, pp. 1–19, Jan. 1924, doi: 10.1086/360550.

[79] “ChatGPT.” Accessed: Jul. 23, 2024. [Online]. Available: https://openai.com/chatgpt/

[80] “Llama 3.1,” Meta Llama. Accessed: Jul. 23, 2024. [Online]. Available: https://llama.meta.com/

[81] “Meet Claude Anthropic.” Accessed: Jul. 23, 2024. [Online]. Available: https://www.anthropic.com/claude

[82] “Gemini - chat to supercharge your ideas,” Gemini. Accessed: Jul. 23, 2024. [Online]. Available: https://gemini.google.com

[83] M. AI, “Large Enough.” Accessed: Jul. 27, 2024. [Online]. Available: https://mistral.ai/news/mistral-large-2407/

[84] Y. Liu et al., “Summary of ChatGPT-Related research and perspective towards the future of large language models,” Meta-Radiol., vol. 1, no. 2, p. 100017, Sep. 2023, doi: 10.1016/j.metrad.2023.100017.

[85] B. Yuan, J. Panneerselvam, L. Liu, N. Antonopoulos, and Y. Lu, “An Inductive Content-Augmented Network Embedding Model for Edge Artificial Intelligence,” IEEE Trans. Ind. Inform., vol. 15, no. 7, pp. 4295–4305, Jul. 2019, doi: 10.1109/TII.2019.2902877.

[86] S. Siriwardhana, T. Kaluarachchi, M. Billinghurst, and S. Nanayakkara, “Multimodal Emotion Recognition With Transformer-Based Self Supervised Feature Fusion,” IEEE Access, vol. 8, pp. 176274–176285, 2020, doi: 10.1109/ACCESS.2020.3026823.

[87] Y. Sun, D. Cheng, Y. Chen, and Z. He, “DynamicMBFN: Dynamic Multimodal Bottleneck Fusion Network for Multimodal Emotion Recognition,” in 2023 3rd International Symposium on Computer Technology and Information Science (ISCTIS), Jul. 2023, pp. 639–644. doi: 10.1109/ISCTIS58954.2023.10213035.

[88] V. G. Morelli, M. P. Barbato, F. Piccoli, and P. Napoletano, “Multimodal Fusion Methods with Vision Transformers for Remote Sensing Semantic Segmentation,” in 2023 13th Workshop on Hyperspectral Imaging and Signal Processing: Evolution in Remote Sensing (WHISPERS), Oct. 2023, pp. 1–5. doi: 10.1109/WHISPERS61460.2023.10430788.

[89] “Home,” OpenCV. Accessed: Jul. 23, 2024. [Online]. Available: https://opencv.org/

[90] “Torchaudio Documentation — Torchaudio 2.3.0 documentation.” Accessed: Jul. 23, 2024. [Online]. Available: https://pytorch.org/audio/stable/index.html

[91] B. H. A. Khattak et al., “A Systematic Survey of AI Models in Financial Market Forecasting for Profitability Analysis,” IEEE Access, vol. 11, pp. 125359–125380, 2023, doi: 10.1109/ACCESS.2023.3330156.

[92] Q. Ding, S. Wu, H. Sun, J. Guo, and J. Guo, “Hierarchical Multi-Scale Gaussian Transformer for Stock Movement Prediction,” in Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, Yokohama, Japan: International Joint Conferences on Artificial Intelligence Organization, Jul. 2020, pp. 4640–4646. doi: 10.24963/ijcai.2020/640.

[93] S. M. Mirjebreili, A. Solouki, H. Soltanalizadeh, and M. Sabokrou, “Multi-Task Transformer for Stock Market Trend Prediction,” in 2022 12th International Conference on Computer and Knowledge Engineering (ICCKE), Nov. 2022, pp. 101–105. doi: 10.1109/ICCKE57176.2022.9960122.

[94] A. G. Lemaître, “A Homogeneous Universe of Constant Mass and Increasing Radius accounting for the Radial Velocity of Extra-galactic Nebulæ,” Mon. Not. R. Astron. Soc., vol. 91, no. 5, pp. 483–490, Mar. 1931, doi: 10.1093/mnras/91.5.483.

[95] E. Hubble, “A relation between distance and radial velocity among extra-galactic nebulae,” Proc. Natl. Acad. Sci., vol. 15, no. 3, pp. 168–173, Mar. 1929, doi: 10.1073/pnas.15.3.168.

[96] R. W. Hafer and S. E. Hein, The Stock Market. Bloomsbury Publishing USA, 2006.