Probing as a Technique to Understand Abstract Spaces

Ashlen A. Plasek
plase024@morris.umn.edu
Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA

Abstract

Machine learning models, while very powerful, have their operation obfuscated behind millions of parameters. This obfuscation can make deriving a human meaningful process from a machine learning model very difficult. However, while the intermediate states of a machine learning model are similarly obfuscated, using probing, we can start to explore looking at possible structure in those intermediate states. Large language models are a prime example of this obfuscation, and probing can begin to allow novel experimentation to be performed.

Keywords: probing, machine learning, large language model, classifier

1 Introduction

Machine learning as a whole often involves working with representations of data which are opaque and unintuitive. Machine learning models are often employed to step from one known form of information (for example, a sentence written in natural English) to another known form of information (for example, that same sentence translated to French) when there is no clear path between the two.

Probing, the technique outlined in Section 3, applies machine learning to aid in understanding the opaque and unintuitive representations of data by finding the relationships which underlie those representations. We will look at two surveys of applications of machine learning to natural language processing which make use of probing to explore the efficacy of the particular models used for embedding. Then we will look at a meta-analysis which exposes some of the methodological issues in using probing to make quantitative statements about particular structures existing in internal representations of data. Finally, we will look at ways probing can be utilized to perform experiments on complex or opaque systems.

2 Background

2.1 Linear Algebra

An ordered collection of numbers can be called a vector. The number of elements comprising a vector is known as the dimensionality of that vector. For example, it is quite common to work with two-element vectors to represent points on a plane, a two-dimensional object. Similarly, a three-element or a three-dimensional vector can be used to represent points in three-dimensional space. We can analogously extend this idea to higher dimensions. Discussing the space of possible vectors of a given dimensionality using spatial analogies is a very powerful tool. One key fact about vector spaces is that elements within them can be added, subtracted, and scaled up or down, just as their two or three dimensional analogs (Equation 1).

$$\begin{bmatrix} 1 \\ 4 \\ -5 \end{bmatrix} + 2 \begin{bmatrix} 3 \\ -8 \\ 3 \end{bmatrix} = \begin{bmatrix} 7 \\ -12 \\ 1 \end{bmatrix} \tag{1}$$

Linearity is a property of a mathematical transformation which satisfies two useful properties. Specifically, the result of summing the inputs and then applying the transformation is the same as applying the transformation to the inputs and then summing (Equation 2), and scaling the input will scale the output by the same amount (Equation 3).

$$f(x + y) = f(x) + f(y) \tag{2}$$

$$f(ax) = af(x) \tag{3}$$

Transformations which take in a vector of a particular dimensionality and produce a vector of another dimensionality can also be linear. In this case, there is a useful property that linearity gives, it is possible to deduce information about the input vector space from the results of applying the transformation to a few key vectors. This will be useful whenever a potentially opaque vector space must be operated on (starting in Section 3). Finally, matrix multiplication provides a useful representation of linear transformation between vectors of potentially varying dimensionalities.

2.2 Machine Learning

Any machine learning model can be viewed as a transformation which, informed by a set of parameters, takes an input and produces an output. Such an abstract representation is not always useful, and a couple of concrete examples will be used throughout this paper. However, taking this abstract view allows us to describe the training process in a similarly abstract manner, giving a necessary perspective for understanding probing.

The process of training a machine learning model entails taking a collection of inputs and a matching collection of the expected outputs for each input, as a whole, referred to as the data or data set. The data set will then be split into

two subcollections, one will be used to train the model, and the other to evaluate its performance after training. To train the model, an input-output pair is selected from the training data-set, and the transformation is applied to the input. The resulting output is then compared with the expected output, and their difference is used to modify the parameters of the model. This is then repeated until a sufficient training period has elapsed or a certain accuracy is met. After training, the model can be tested by taking input-output pairs from the evaluation set, and the transformation given by the trained set of parameters is applied to the input. The resulting output is compared with the expected output, and the difference is evaluated in some way. A function called a loss function is used to take this difference and produce a numerical value representing the severity of the error [3].

2.2.1 Linear Classifiers. To be more precise about how a transformation takes place, we will examine linear classifiers. Like all machine learning models, the linear classifier applies a transformation to an input and produces an output. In the case of a linear classifier, the transformation can be represented as the linear transformation given by a matrix of an appropriate size to take the input vector space to the output vector space. The values comprising the matrix are the parameters of the model, and applying the transformation is applying matrix multiplication to the input vector to produce the output vector.

As the name suggests, a common application of a linear classifier is to take a set of inputs and determine which of a series of categories best describes that set of parameters. This is often expressed as giving each entry in the output vector a category, and whichever entry is the greatest determines the classification produced by the model.

2.2.2 Neural Networks. The usefulness of matrix multiplication does not stop with linear classifiers, and there are a number of modifications which can be made to produce a more complex machine learning model. The first key modification that can be made is to augment the transformation to introduce non-linearity. While this removes the ability to reason as easily about the relationships between subsets of the input vector space and the output vector space, it is often more useful to have more complex (non-linear) regions represented in the output space. To achieve this, a variety of non-linear functions (called activation functions) can be applied to each component of the output vector to introduce non-linearity. Some activation functions can preserve linearity as much as possible, others are discontinuous, whereas others are smooth, continuous functions such as the ubiquitous sigmoid curve (Figure 1).

By introducing non-linearity, composing multiple transformations becomes a useful prospect. Whereas with matrix multiplication, where successive applications of transformations can be represented by a single transformation

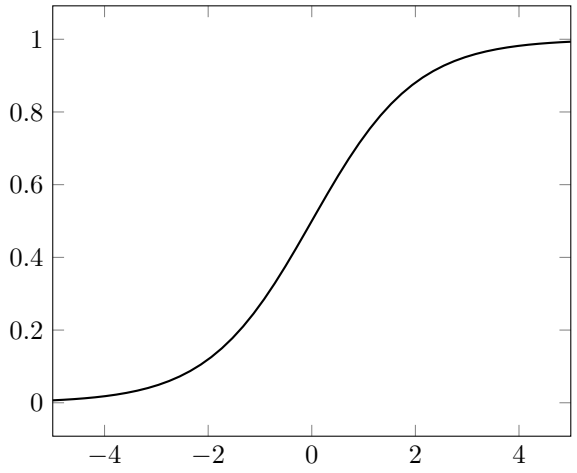

Figure 1. An example of a sigmoid curve

(and therefore a single matrix multiplication); adding non-linearity makes the composition more complex. This use case makes composing many layers of matrix multiplication followed by the application of an activation function a powerful tool for constructing more complex models. In addition to increasing the complexity of the regions which the model can operate on. Increasing the number of parameters can increase training speed by allowing more potential representations of a given region.

2.2.3 Internal State and Large Language Models. While a full exploration of how Large Language Models work and the various ways they can be implemented and trained is beyond this paper's scope, some features are necessary to address. The most important is that most large language models take their input as a stream of fragments of text and generate one fragment of their output at a time (for example, a word or phrase). This is done by using an internal state used to keep track of information about where the model is in the output being generated [8]. While not necessarily representative of any given model, this can be understood as allowing some of the inputs to a neural network to be a portion of the outputs from the previous step. This creates a feedback loop, allowing the model to possess information about where in the output stream it is. The portion of the output given as input would be the internal state. Just like all of the other information passed through machine learning models, the internal state is a vector living in a vector space.

3 Word Embeddings

Natural language processing is a subset of computer science which is concerned with interpreting, transforming, and outputting natural language. Natural languages such as English or French can be contrasted against structured languages such as the programming languages C or Haskell in that natural languages have a far less rigid structure and are far

$$\begin{bmatrix} 118 & \text{v} \\ 101 & \text{e} \\ 99 & \text{c} \\ 116 & \text{t} \\ 111 & \text{o} \\ 114 & \text{r} \\ 0 & \\ \vdots & \\ 0 & \end{bmatrix}$$

Figure 2. ASCII-based embedding of the word "vector"

$$\begin{bmatrix} 0 & a \\ 0 & Aalenian \\ \vdots & \\ 0 & vectitory \\ 1 & vector \\ 0 & vectorcardiogram \\ \vdots & \\ 0 & zythum \\ 0 & Zyzzyva \end{bmatrix}$$

Figure 3. Simple embedding for the word "vector"

more difficult to parse using a concrete grammar. Due to this lack of rigidity, natural language processing is an excellent application of machine learning, specifically because pairs of inputs and expected outputs can be used to train the model instead of developing a complex grammar to attempt to parse natural languages.

3.1 Words as Vectors

To apply machine learning to problems in natural language processing, the inputs and outputs must be converted into a vector representation. For example, in machine translation, a string of text must be processed into another string of text in another language. Thus, a process for converting a string of text, in this case a word, into a vector to represent that word is needed. We will examine three methods of doing this, a very simple method based on character encodings, an effective but inefficient technique using one dimension per word, and finally, a simplified version of a popular technique used in the field.

3.1.1 Character Encoding. If a word were to be interpreted only as a sequence of numbers representing each character, one possible solution would be to simply use the character encoding of text to produce each value in a vector. Figure 2 shows the word "vector" in a vector representation. There are a number of problems with this representation. The most apparent problem in choosing a dimensionality for the input and output vectors. The dimensionality will place a cap on the number of characters allowed in the input. But the more pressing problem is in thinking of these embeddings as points living in a space, very dissimilar words could have very similar representations. For example, "brand" and "bread" are very similar in vector space but represent very different concepts.

3.1.2 Higher Dimensionality. Take a vector with a single dimension for every word in a language. The vector for a given word would have a 1 in the entry for itself, and a zero in every other entry. This is demonstrated for the word vector in Figure 3. For English, a conservative estimate places this at around 500,000 words, with around 1,300,000 distinct definitions [7]. While this is a very high dimensionality, it gives plenty of space for a machine learning model to generate structure within when processing such a vector. However, this high dimensionality produces a problem: there is a clear lack of efficiency in working with these vectors and there is no encoding of the meaning of the original word.

3.2 Encoders and Decoders

With a simple but inefficient embedding produced, the next problem to solve becomes reducing the number of dimensions in the vector. A slightly more complex application of machine learning makes this possible without needing to produce the structure within the desired smaller vector space by hand. To achieve this, construct a neural network by composing two layers. The first layer maps a vector in a 1.3 million dimensional space down to a vector in a 500-dimensional space, and the second layer maps a vector in a 500-dimensional space to a 1.3-million dimensional space. This is shown in Figure 4.

This model can then be trained on a dataset consisting of every word being mapped to itself. On the surface, this is just training a machine learning model to apply the identity function. However, if the two layers are separated, there is a transformation which is able to compress the 1.3 million dimensions of the input space down to just 500, and there is a second transformation which can take the 500-dimensional vector, and map it to a 1.3 million dimensional vector. In other words, a word can be embedded into a 500-dimensional-space, and then taken back out to recover the original word.

This is a very simple model to produce a word embedding, and more complex training techniques can be employed to impose structure onto the embedding. It is important to note, however, that any imposed structure cannot be precisely controlled. One method of producing a more structured embedding is to provide context for the words being embedded; such a technique may impose structure representing semantic meaning of words.

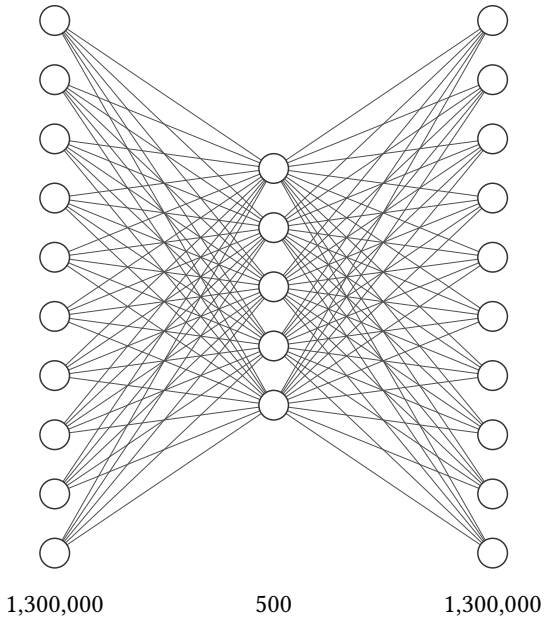

Figure 4. Encoder decoder pair

3.3 Probing for Performance

There is a limit to the efficiency gains in embedding words into a lower-dimensional vector space. It is clear that a single number could not represent all of the structure and semantic information to be had in a word, however, it is hypothetically possible for an encoder-decoder pair to produce such an embedding by mapping 1,300,000 distinct regions along the real number line to each of the possible words. Doing so would lead to a problem similar to that encountered in the character encoding embedding, the structure of the vector space is not being used efficiently.

To remedy this, the notion of a property being represented must be stated in a more precise fashion, specifically that there are properties of the words which are not able to be extracted reasonably from the embeddings. To test if a given property (for example part of speech) is present in the embedding, machine learning can be applied in a more novel way. Instead of using training to produce a model, the accuracy estimation is the result. If a simple model, such as a linear classifier, is applied to an embedding, and trained with a dataset of words tagged with their parts of speech, the accuracy achieved via training can be used to interpret the presence or absence of a property.

For example, if we wanted to test if the property that a word is a noun is represented in an embedding, we could train a linear classifier on embedded words (their vector representations) expecting a single value, representing if that word is a noun or not. If the linear classifier can successfully model such a distinction, there is some direction in the embedded space which represents if a word is a noun.

Machine learning models are traditionally trained to complete a task, and the accuracy of the model on the testing data set is used as a predictor of the performance of the model in the task assigned. The results of the training process in such a case are the parameters which allow the model to perform its task. Probing differs from this methodology by using the accuracy of training not as an evaluation but as a measure of how accurately a particular property is represented in the embedded space.

Köhn made one of the early explorations into probing by comparing the performance of five embedding algorithms by using probing to detect the presence or absence of several properties, including the part of speech, tense, gender, count, and number of the word [4]. On each of these tasks, the five embedding algorithms were evaluated on seven languages.

When performing their evaluation, Köhn used a linear classifier for probing. Explaining that, "the feature dimensionality is relatively high ... and more importantly, training a linear classifier yields insights into the structure of the vector space because the classifier also serves as a tool to obtain a supervised clustering of the vector space"[4]. In other words, because the region which maps to a given class can be determined due to the linearity of the model, it becomes possible to find patterns in the input space, potentially revealing information about the space formed by the embedded vectors.

Among the observations made by Köhn, the dimensionality of the embedded vectors was the most important factor in improving accuracy, noting that while 200 dimensions does offer a benefit over 100 dimensions, it is far lesser than the increase from 10 dimensions to 100 dimensions, as such it is possible to pair such analysis with an evaluation of the task the embedding is being used for to find the best dimensionality for that use case.

4 Sentence Embeddings

A similar methodology can be used to embed sentences as vectors. The core methodology is similar and often involves an encoder-decoder pair, but training can include modifications to encourage particular properties to be reflected in the embedding, although there is the desire for "'universal embeddings' trained once and used in a variety of applications" [2]. Such an embedding would ideally be reflective of a vector space which encodes any useful property a sentence embedding could maintain. Because it is impossible to enumerate all possible sentences which can be constructed, it is impossible to exhaustively train an encoder-decoder pair thoroughly enough to embed all possible properties.

4.1 Properties of Sentences

There are many such properties that can be desired in a sentence embedding. Conneau's investigation of many encoders in the field of sentence embedding by testing their resulting

embeddings for the presence of a set of properties. These properties are divided up into three broad categories which approximately correspond to the difficulty in representing a property [2]. The three categories are surface properties, syntactic properties, and semantic properties.

4.1.1 Surface Properties. Surface properties of a sentence are the simplest set of properties, and do not require linguistic analysis to determine. For example, the length of a sentence is a surface property. A more involved property could be if a given word is present in the sentence. While this requires processing what constitutes a separate word, there is still no semantic analysis to be done [2].

4.1.2 Syntactic Properties. Syntactic properties are the next simplest, and require some linguistic analysis to determine the structure of a sentence [2]. For example, determining if a sentence is syntactically correct. This does not imply the sentence has meaning, however, only that words with the right parts of speech appear in the proper order for the sentence to be interpreted as a syntax tree. Another example of such a property involves a tree-based representation of a sentence, and is the height of that tree. This tree is generally a formalization of the process of diagramming a sentence.

4.1.3 Semantic Properties. Finally, semantic properties are the most complex and require interpretation of the meaning of sentences [2]. For example, recognizing the difference between a sentence that is syntactically correct but semantically meaningless and a sentence which is semantically meaningful as well as syntactically correct. Another example would be determining the number of subjects of the sentence or the verb tense of the main clause.

4.2 Probing Analysis

Conneau determined that, "performance is still far from the human bounds ... [for determining] if a sentence is syntactically or semantically anomalous"[2], however also noted that the poor performance could be correlated with the encoders not being explicitly trained to determine semantic or syntactic information. This indicates that not only is probing capable of determining the presence or absence of a property, it can give an approximate measure of the ability for a given embedding to enable processing of particular properties or types of knowledge.

5 Criticisms

In 2022, Belinkov compiled a review of the work done in probing, specifically looking at some of the shortcomings of probing as a technique in evaluating language models. A number of problems are noted, including the lack of a consistent baseline, the use of different models for classification tasks, and determining if a classifier's accuracy is a result of correlation or causation [1].

5.1 Lack of Comparison

The lack of a baseline or a control is a particularly troubling problem, as while it may appear to have a solution, the solution introduces methodological questions. Belinkov notes, "Some studies compare with majority baselines or with classifiers trained on representations that are thought to be simpler than what the original model f produces"[1].

However, Belinkov also observes, "On the other hand, some studies compare [performance] to skylines or upper bounds $\overline{f}$, in an attempt to provide a point of comparison for how far probing performance is from the possible performance on the task of mapping [an embedding to a particular feature]"[1]. In other words, using a shortfall in performance from another example to demonstrate a lack of capability, a metric which is difficult to interpret without a control, which is a commonly lacking feature in probing analysis.

This reveals a meta-analysis problem as the method of evaluation varying from one body of research to another makes it difficult to build upon other work without replicating the baselines used to match against a control.

5.2 Choice of Model

When choosing a model to serve as a probe, researchers inadvertently influence their results as using a more complex model can reflect a perceived complexity of the classification problem at hand, even when the same probing model is used for many classifications. While broadly, Belinkov found that different models used for probing can yield usable information to compare the abilities of two different models, it is still possible for performance comparisons to be negated by such differences [1].

5.3 Correlation and Causation

The primary problem Belinkov addresses is that it is difficult to determine if the success of a classifier is due to a genuine causative connection between an embedding and the tested property, or if there is simply a correlation between a property the encoder was trained to note and the property being tested for [1].

For example, if we wanted to determine if a particular embedding neatly represents if a given word is plural, we could apply a probe for that property and get a success. However, consider the property that a given word ends in an 's'. Because there is a high correlation in English between nouns that end in an 's' and words that are plural, it is possible that the success we observe is instead due to this correlation and not due to a proper representation of plurality.

Belinkov goes on to note two proposed solutions. Both involve, on some level, modifying the encoder to determine more precisely if a property is present in an embedding. The first is to modify the original encoder while training the probing model, and then assess both the performance of the probing model, and the performance of the modified

encoder on the original task, thus allowing a causative link to be established. The second method is more direct. By training a linear classifier to determine a given property, it is possible to use that classifier to remove that property from the embedding, allowing evaluation in the absence of a potentially interfering factor[1].

This removal is performed by determining what subset of the embedding space is associated with the interfering property and collapsing the embeddings under test so as to remove any component of that property from the embeddings, thus removing the possibility of interference from inference.

5.4 Other Options

It should also be noted that probing is not the only method of evaluating the performance of NLP systems. Work done by Google has produced a large benchmark suite for language models which aims to provide a robust, future-proof metric for language models [6]. However, this is a different objective than the evaluation probing can accomplish. Probing is most effective at evaluating the encoders used for word or sentence embedding, and falls short when evaluating the competency of language models as a whole.

6 Wider Applications

The method of separating correlation and causation given by Belinkov unlocks the ability to not just understand the space of embeddings, but to modify vectors living in the space of embeddings. This can be with the intention of removing properties as in the method proposed by Belinkov, or to modify vectors living within that space for the sake of experimentation.

Li et al. make use of this technique to modify the internal state of a large language model. The researchers question if large language models have an internal representation of the world, focusing on the example of a language model trained to play the game Othello. Othello is a board game played with black and white tokens on a grid, similar to Go in appearance but significantly simpler. The model was trained based on a dataset of games where moves are given in sequence. As expected, the model performs very well once trained, achieving an error rate of 0.01% when trained on an artificial dataset designed for training, and 5.17% when trained on a dataset of championship games [5]

Probing is then used to determine if the model has an internal representation of the board state. Li et al. start with testing linear probes, but that approach produced an error rate exceeding 20% when classifying if a given tile contains a colored token or is empty. As such, they switch to a 2-layer neural network with an activation function chosen to maximize linearity. Such a model greatly improved error rates, bringing the error rate down below 12%, with some layers of the internal state giving error rates as low as 2 to 3% [5].

With a model trained to determine board state, and a model which maximizes linearity, it is possible to reasonably determine the representations of the various board positions. In this way, it becomes possible to modify the internal representation of the board state. Thus, the model can be tested for its ability to produce valid moves in novel scenarios which it could not reach through either its training data or the games it could generate. Further, comparing the outcomes of these experiments for the model trained on artificial data and that trained on championship data reveals that the championship dataset produces a model which "make[s] strategically good moves." [5] Li concludes by comparing the probability maps of the next moves produced by the model pre- and post-intervention to conclude that the large language model does have a representation of the board state.

7 Conclusion

Machine learning is a multitool, not only in its ability to solve problems in a variety of domains, but in the results the training process can reveal. By shifting perspective from one of attempting to use machine learning to directly solve a problem, but instead, to use the results of training to extract information about the properties of embedded vectors, it becomes possible to break down the obfuscation surrounding the methodology with which machine learning models solve problems. This shift in perspective, and a willingness to apply such perspectives to problems in natural language processing led to the development of probing.

Probing both makes it possible to investigate the structure of a potentially opaque word or sentence embedding, but also using those results to inform future research, and better understand the differences between embeddings. Furthermore, by leveraging the linearity of linear classifiers, it becomes possible to understand the structure imposed on the vector space represented by embeddings far more effectively than simply testing inputs to an encoder.

Finally, knowledge of the vector space embedded objects live in makes it possible to explore that space more literally, allowing modification of vectors in that space to be done via an evidence-based method, thus allowing experimentation. Applying these tools beyond embeddings, to the broader world of large language models, it becomes possible to learn about the internal state of a generative machine learning model, and even potentially run experiments to determine what form of processing is occurring within these models.

Acknowledgments

I would like to thank Peter Dolan, my adviser for this senior seminar, for the vital feedback and all the help throughout the process of composing this paper. I would also like to thank Elena Machkasova, my professor, for all of her essential

feedback and help guiding me through this process. Lastly, I would like to thank my alumni reviewer for his detailed feedback.

References

[1] Yonatan Belinkov. 2022. Probing Classifiers: Promises, Shortcomings, and Advances. *Computational Linguistics* 48, 1 (04 2022), 207–219. https://doi.org/10.1162/coli_a_00422 arXiv:https://direct.mit.edu/coli/article-pdf/48/1/207/2006605/coli_a_00422.pdf

[2] Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Melbourne, Australia, 2126–2136. https://doi.org/10.18653/v1/P18-1198

[3] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning* (12th ed.). Springer, 18–19.

[4] Arne Köhn. 2015. What's in an Embedding? Analyzing Word Embeddings through Multilingual Evaluation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, 2067–2073. https://doi.org/10.18653/v1/D15-1246

[5] Kenneth Li, Aspen K. Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2022. Emergent World Representations: Exploring a Sequence Model Trained on a Synthetic Task. (2022). https://doi.org/10.48550/ARXIV.2210.13382

[6] Aarohi et al. Srivastava. 2022. Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models. https://doi.org/10.48550/ARXIV.2206.04615

[7] Wiktionary. 2023. Statistics — Wiktionary. https://en.wiktionary.org/wiki/Special:Statistics. [Online; accessed 01-March-2023].

[8] Stephen Wolfram. 2023. https://writings.stephenwolfram.com/2023/02/what-is-chatgpt-doing-and-why-does-it-work/