Teleosemantics for Neural Word Embeddings

A theory of the representational content for a system should satisfy two criteria at a minimum. It should *explain* the role that content plays in our lower-level explanations of the operation of the system. In the case of 'interpretable' or 'explainable' AI, a theory of content should tell us what it is we are trying to explain or interpret in the first place. It should shed light on questions like, how did the system come to have representational content? What kind of content can the system represent? At the same time, it should *constrain* the contents we ascribe to the relevant system. For Large Language Models (LLMs) this question has additional significance. Human interpreters are prone to ascribe a text the full representational significance it has in their own idiolect and, as a result, there is a considerable risk that they will over-attribute content to the text produced with the use of language models. It is not unreasonable to think that the contents ascribed to texts generated with LLMs should be constrained by the representational capacities of these models (though this is not the only approach to interpretation, Mallory, 2022, 2023). At a time when the representational capacities of large language models are being drastically overstated (De Cosmo, 2022), the application of a general framework for distinguishing genuine from spurious representations can provide a bit of perspective (and cold water). This is where teleosemantics comes in.

Teleosemantics is one of the most promising naturalistic theories of what makes something a representation. It explains the category of representation in terms of *function;* where a function is an effect which has been selected by an evolutionary process. The core idea is that certain states have evolved to represent how things stand in the world because cognitive/computational mechanisms have evolved to reliably produce and consume these states. While teleosemantics was not developed to explain the contents of artificial neural networks, it has had some success in this domain (Shea, 2018, Buckner, 2021, Piccinini, 2022). However, it doesn't provide a simple algorithm for determining the semantic content of a representation — the method of 'probing' in machine learning is just as much an empirical task as its analogs in neuroscience (see Ivanova et al. 2021 for a discussion of some parallels). What teleosemantics provides is a relatively precise and general account of how artificial neural networks might utilise representational contents in their computations and a means of determining whether these representations are 'intentional' in a theoretically precise sense. Importantly, for discussions of machine 'intelligence', it gives us a means of distinguishing 'original from a merely 'derived' or 'imputed' intentionality.

This paper will begin the project of applying teleosemantics to neural language models at the base, Tomáš Mikolov's Word2Vec algorithm. The first section will give a quick overview of neural language models and introduce Word2Vec in some depth. The second section gives a quick overview of teleosemantics while the third recasts the operations of the word2vec algorithm within the teleosemantic framework. The fourth section will consider how this meets the two goals for a theory of content identified above; whether it can explain and constrain the content we ascribe to word embeddings. While the main focus of this paper will be on a particular and relatively straight-forward framework of neural language modelling (i.e., no attention heads, no fine-tuning), a secondary aim is to indicate how teleosemantics can be applied to neural networks more generally and so we will have to engage with some current debates in the literature along the way, in particular, debates about the role of intention in determining the function of an artefact, and the individuation of vehicles of representation in neural networks.

Our results will be a mixed bag. While teleosemantics does suggest that Word2Vec embeddings are intentional representations, it does not suggest that they represent the 'meanings' of lexical items. Rather, the embeddings are representations of probability distributions over a lexicon. This is quite a conservative result. However, by shedding a light on the role of structural representation and downstream task-functions, the framework can help to explain why more sophisticated transformer models can come to have different representational contents.

1.1 A brief introduction to word embeddings

Since 2013, the use of neural networks to produce word embeddings has become widespread in the field of natural language processing.[1] Word embeddings are vectoral representations of words (i.e. lists of numbers) which can be used for a range of downstream computations. Traditionally, these vectors would be produced by counting the co-occurrence patterns of tokens in a corpus and representing the number of times two tokens occur together as values of a given dimension. This would produce massive vectors with millions of dimensions which were wholly interpretable in the sense that you would know what each dimension corresponded to, e.g., the number of times two particular words occurred next to each other, but were computationally expensive. The innovation behind neural word embeddings is to produce word vectors by training neural networks on language modelling tasks. The vectors which are produced can be much smaller than traditional word vectors, typically around 300-500 dimensions, but with the cost they become much harder to interpret. We can't simply say what feature a given dimension corresponds to if any.

Importantly, neural word embeddings seem to develop representational properties unfound in traditional word vectors. On a surface level, the embeddings appear to group words with more 'similar meanings' closer together within the vector space (Mikolov et al. 2014). For example, the vector for 'cat' will be closer to the vector for 'dog' than that of 'table' according to a similarity metric like cosine similarity or simply computed as the dot product of the respective vectors. Beyond mere proximity, at a deeper level, the vectors support a surprising degree of analogical reasoning (Ibid). Word vectors, like any vectors, can be added to and subtracted from each other. To use a famous example, subtracting the vector for 'man' from the vector for 'king' and then adding the vector for 'woman', produces a vector whose nearest neighbour is that of 'queen'. Such analogies hold for a wide range of conceptual relations. At another level, it is possible to project vectors into 'semantic subspaces' which capture the placement of words on a scale (Grand et al. 2022). One can draw a line between regions in vector space fixed by sets of antonyms, for example, a line from the middle of the region of space containing {large, big, huge} and the region containing {small, little, tiny}. Using this line as a scale, it is then possible to find a semantic projection of vectors to positions on that line which corresponds to human judgments of scale (see Appendix 1). For example, a 'horse' is larger than a 'cat' which is larger than a 'mouse'. A further possible level of representation emerges if we use dimensionality reduction methods such as Principle Component Analysis (PCA) to collapse the number of dimensions down to 2. For example, when reduced to 2 dimensions, the set of vectors for the names of some capital cities appear to track broad geographic relations between cities. In a sense, one could use this representation as a geographical map of Europe (see Appendix 1. I will call these 'PCA maps').

To sum up, simple word embeddings appear to have several levels of representational structure. The first level is captured by cosine (or dot product) similarity relations between vectors. At the second level, analogies are captured by vector addition and subtraction. The third level corresponds to sets of linear transformations between vectors. The final kind of representation is uncovered by dimensionality reduction (e.g., PCA).

Lately word embeddings have played a central role in the development of transformer-based language models. Transformer models use word embeddings encoded with positional information which are then fed into 'attention heads' which

[1] This is in large part due to the work of Mikolov (Mikolov, 2013), however, the idea of producing embeddings by training a network is older (Collobert & Weston, 2008).

compute the similarity (e.g., dot product) of the embeddings before using this information to produce attention weights indicating how much different embeddings should be influenced by each other (Vaswani et al. 2017). This produces contextualised word embeddings, that is, embeddings whose exact values are sensitive to surrounding words. Unlike the simple embeddings that will be the focus of this paper, contextualised embeddings are able to distinguish ambiguous words (e.g., 'bank' and 'bank') and are claimed to produce better representations of word 'sense' (Peters et al., 2018). These also assist transformer models in the representations of syntactic structure (Lappin, 2021, Linzen, 2018). For the next few sections, I will be setting transformer models and attention mechanisms aside to focus on simple neural word embeddings. But first, we need to look at teleosemantics.

2. A brief introduction to Teleosemantics

A theory of the representational content of neural word embeddings needs to be able to distinguish genuine, endogenous representational contents from putative, or merely attributed contents. It should tell us what content is really present in the embedding and what content is purely imputed. To provide this, we need an account of what makes something a genuine, endogenous content. Teleosemantics grounds its account of content in terms of proper function. The fundamental idea is that some systems have the proper function of producing representations and that this function is a result of how those systems have evolved.

An evolutionary process requires selective reproduction, that is, both stable replication and a method of selection among replicators. The definition of reproduction is somewhat complicated due to the need for it to cover both the reproduction of mechanisms (e.g., the visual system) and the representations that the mechanism can generate (e.g., individual colours). Following Millikan, 1984:

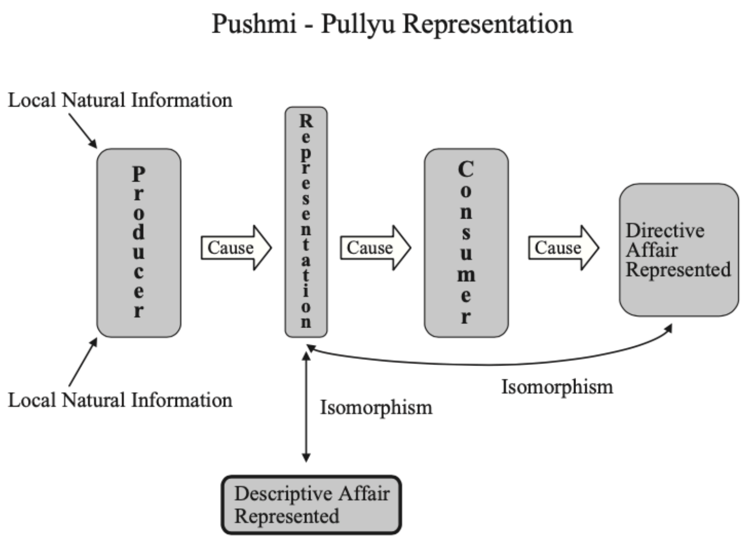
A is a reproduction of B iff

i. Similarity: B has some determinate properties in common with A,
ii. Regularity: That it does so can be explained in terms of some natural law (or natural law in situ), and
iii. Counterfactual sensitivity: For each property p, these laws explain why if A had been different with regard to p (within some range of possibilities), B would have been different with regard to p. In other words, with the direction of causality leading from A to B.

Using this account of reproduction, Millikan's then defines 'proper function' as follows:

> Proper function: "For an item A to have a function F as a "proper function", it is necessary (and close to sufficient) that one of these two conditions should hold. (1) [direct proper functions] A originated as a "reproduction" (to give one example, as a copy, or a copy of a copy) of some prior item or items that, due in part to possession of the properties reproduced, have actually performed F in the past, and A exists because (causally historically because) of this or these performances. (2) [derived proper functions] A originated as the product of some prior device that, given its circumstances, had performance of F as a proper function and that, under those circumstances, normally causes F to be performed by means of producing an item like A" (Millikan, 1989).

The next step is to use the concept of a proper function to illuminate that of representation. According to consumer-based teleosemantics, a producer mechanism has the proper function of producing a state (i.e. a representation) in response to some external state of affairs. This state is then used by a consumer mechanism which has the proper function of performing some behaviour(s) which may only be successful when


Pushmi - Pullyu Representation
Local Natural Information
Producer
Cause
Representation
Cause
Consumer
Cause
Directive Affair Represented
Local Natural Information
Isomorphism
Isomorphism
Descriptive Affair Represented

that external state of affair occurs.[2] In the simplest kind of system, the content of a representation is both descriptive and directive — the consumer system *must* perform some task in response to it. The descriptive content is the state of affairs which prompted the producer system to produce its representation while the directive content is the state of affairs in the world which the consumer system is to bring about. Millikan calls these 'pushmi-pullyu representations' (PPR). A bee's waggle dance indicates both where nectar is and where other bees are to go. A rabbit's hind leg thumps indicate both that danger is near and that other rabbits are to freeze or hide. A neural signal can indicate both that an object is hot and that a body part must be immediately withdrawn [check this].

The content of a descriptive representation is given by the environmental conditions which were in place such that the consumer system's behaviour lead to its survival and reproduction. To take a classic example, when a frog's visual system (the producer) detects a small, black moving object and produces a signal which results in an assembly of motor mechanisms (the consumer) sending its tongue out in the direction of the object, we ascribe an evolutionarily appropriate content to that signal. The signal would have contributed to the survival and reproduction of the frog if the small object were a fly but not done so if the object were a shadow and so the appropriate content to ascribe the signal sent from the visual system to the motor system is that there is a fly present.

Since we will be interested not just in the content of individual representations but in relations between representations (e.g. 'semantic similarity'), we need to also take into account 'structural representation' (Swoyer 1991, Ramsey, 2007). For example, if the map above genuinely represents the cities of Europe, it does so in virtue of the relations between its points; it is a structural representation. A structural representation is 'A collection of representations in which a relation between representational vehicles represents a relation between the entities they represent' (Shea, 2014).

The category of structural representation rises distinct problems. There may be many possible mappings, gruesome and otherwise, between the states of the system and the environment in which it finds itself, but not all of these mappings are exploited by the system to perform its task. To use the classic example, the angular separation between two distinct honey bee dances (in the same location) will indicate the angular separation between two different sources of nectar. As such, there is a structural relationship between the set of dances and the geometric properties of the environment.

[2] Alternatively phrased, the state of affairs which prompts the production of a representation must feature in a Normal explanation of how the consumer system performs its task.

However, there is no evidence that these relations between dances are exploited by the bee's consumer systems to map their environment and so it would be inappropriate to treat these structural relations as the content of the dance. Shea proposes the following condition on isomorphisms:

> Exploitability: "(i) A 1-1 map between a set of putative representational vehicles in an organism or other system, and a set of entities to be represented, (ii) in which a projectable relation on the vehicles to which operations of the system could be sensitive (iii) corresponds to (iv) a relation, of significance to the system, on the entities to be represented" (Shea, 2014).[3]

This exploitability criterion is important for constraining the liberality of the notion of structural representation. Without it, there may be any number of mappings by the (evolutionarily fixed) internal states of a system/organism and objects in their environment. Exploitability provides some constraint on this.

3. A brief introduction to Word2Vec

In this section, I will present in some detail, the operation of the word2vec algorithm. [4]We'll start with a discussion of how the algorithm generates outputs from a given input and how the back-propagation algorithm functions before discussing a particular training method, the skip-gram model.

Word2vec uses a shallow, 3-layer neural network to produce dense vector representations for the words in a dataset. The first layer, *input*, takes the input from the dataset, from this input, the second layer, *h*, represents the embedding which the network learns during training, and the third layer, *output*, gives the output of the task the network is trained to perform. In the process, it applies a softmax operation which I

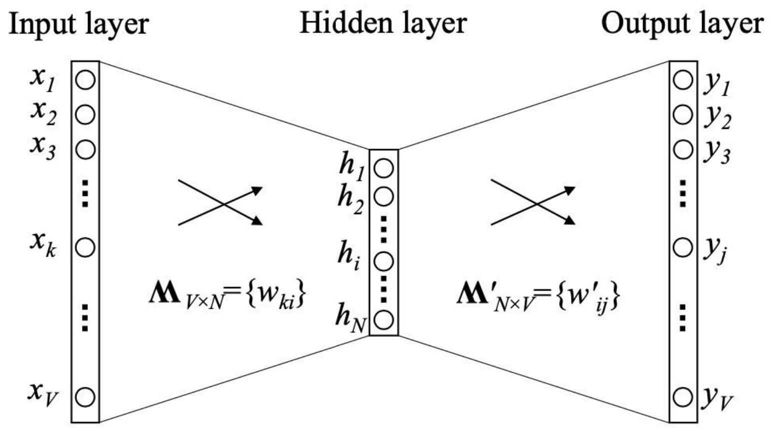


Figure 1: A simple CBOW model with only one word in the context

will discuss later. There are two general ways of producing word embeddings with word2vec. The first involves taking an individual word as input and having the network

3 Shea 2018 weakens this to 'exploitable structural correspondence'. "An exploitable structural correspondence is a structural correspondence between relation V on vehicles v_m in a system S and relation H on entities x_n in which (i) V is a relation that processing in S is systematically sensitive to; and (ii) H and x_n are of significance to S" (Shea, 2018). The relation is of 'significance' to S if it influences how outcomes are robustly produced and stabilised.

4 This account is heavily influenced by the work of the late Xin Rong (Rong, 2014).

guess which words occur in its surrounding context (i.e. continuous bag of words). The second method takes contexts as inputs and has the network guess what words are likely to appear in those contexts (i.e. skip-gram).

In the following, I will describe a simplified version of the training algorithm which, when fed an input word, *w*, seeks to produce a probability distribution over possible context words. The primary focus during training will be the connections between the layers. Call the connections between the input and hidden layer, M (M for matrix, M for mechanism) and the connections between the hidden and output layer M′, the set of words in the training data will be called V (V for vocabulary). M and M′ are sets of weights and biases which determine the function the network computes. Since they are, in effect, a structured set of numbers, we can view them as matrices. At the start of training, the matrices M and M′ have been randomly initialised. During training, the values of M, the weights and biases, will change to produce embeddings for words in V which are useful for completing the task carried out by the second matrix, M′. M′ takes an input from h and produces a probability distribution over elements in V indicating their likelihood of occurring in a context with *w*.

Now for some more detail. The input layer has |V|-many nodes (where |V| is the size of the vocabulary of the dataset) and it provides a one-hot encoding of each word type in vocabulary. This encoding can be viewed as a vector v_{in} with the value 1 for one unique dimension for the word and 0 for all |V|-1 other dimensions (e.g., [0, 0, 0, 0, 1, 0, ….0]). This is a 'representation' of the word, in a sense, there is a unique vector for each word, but it contains no richer content. The first transition in the system multiplies the input vector, v_{in}, by the matrix M. Because it is a one-hot encoding, the effect is simply to select the row of the matrix which is not 0 and copy it to the hidden layer *h.* While the input layer is |V|-many nodes, the hidden, embedding layer is N.

$$h = M^T v_{in}$$

The consumer matrix M′ takes h as its input and produces an output vector, again computed by simple matrix multiplication.

$$v_{out} = M'^T h$$

Recall that M′ is a N X V matrix and so the output layer has the same number of nodes as the vocabulary. This makes it possible to interpret the values of the vector v_{out} as scores for each word in the vocabulary, indicating their likelihood of co-occurring with the input word. To do this, we apply a softmax function to the vector, a non-linear function which ensures that the values in the vector sum to 1 (fig 3.). It is frequently asserted that the softmax function computes a probability distribution, in this case, a distribution over the words in the vocabulary. It is absolutely vital to note that this is a semantic claim about how the outputs of the function should be interpreted. The mathematics does not compel us to endorse this or any other interpretation of the function. One thing which we should hope a theory provides us with is a justification for this interpretation, that is, an explanation for why we are justified in interpreting the outputs of the model as a probability distribution.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

fig 3. (Softmax function)

The output distribution is then used as the input for the back-propagation algorithm. This algorithm compares the output distribution with the actual co-occurrence patterns in the training data, compute how far off the network was (i.e. error), and then update the model parameters accordingly. The training objective is to maximise the probability of seeing the actual target word, w_T, given a particular input word w_I. We can think of this in terms of a probability distribution over possible output words. The loss function tells us how far the output distribution is from the target distribution. Specifically, it gives us the negative log probability that one would see the target word given the input, $E = -\log p(w_T|w_I)$. The aim of the backpropogation algorithm is to minimise the output of E. The method is to determine, for each weight in the network, how sensitive the output of this function is to slight changes in that weight, and then to use that information to change the weights to more effectively minimise the loss. To do this, the algorithm subtracts the derivative of the loss function at a given weight from that weight's value to generate a new weight. The greater the derivative (i.e., the rate of change) at that input, the more the weight is changed. As the algorithm works on the weights in both M′ and M it produces embeddings which more effectively minimise loss while also generating a unique structure for the consumer system M′. Focussing on M′, the update function for the backpropogation algorithm is given by the following equation (the exact details don't matter as much as the fact that we can determine exactly how much loss impacts change):

$$\text{Update rule: } w_{ij} = w_{ij} - \eta \cdot e_j \cdot h_i$$ [5]

There is no guarantee that this algorithm will result in a global minimum, that is, the combination of weights and biases that produces the lowest possible level of loss, but it will reach a local minimum. As it does this, both the set of vectors corresponding to the words in V and the internal structure of M′ will change to more successfully (with less loss) complete the word prediction task.

The method presented indicates how the algorithm produces a probability distribution over words in response to a single word input. In practice it will use one of two standard training methods. When running a CBOW (continuous bag of words) model, the network will actually take several words as input, the exact number to be set as a hyperparameter C, and the hidden layer will be determined by the product of matrix, M, with the average of the input vectors (since we're using averages, word order doesn't matter). Alternatively, the Skip-Gram model takes a single word as an input and instead of producing a single distribution, it will produce |C| distributions. In effect, the Skip-Gram model attempts to predict the context of a word when given that word as input. The differences between these approaches won't actually matter for our analysis later.

4. Applying Teleosemantics to Word Embeddings

4.1. Defence of the Method

Before applying the teleosemantic framework to Word2vec, it's necessary to respond to a criticism of this project. The three core ideas behind these criticisms are that the proper function of an artefact is fixed by the intentions of its designers, the function for which a language model has been designed is word prediction, and the endogenous representational capacities of an artefact cannot overstep this intended function. (a version of this argument can be found in Butlin, 2021).[6] If correct, this argument renders it a priori that language models can only represent probabilistic relations between words since this is what they were intended to do. Even if, during training, they appear to track other properties implicit in the data set, the question of what they represent was answered before training began.

[5] η denotes the learning rate and e_j is the value got by subtracting the target value of word j from the network's output value. Technically, $e_j \cdot h_i$ is the derivative of the loss function at the weight w_{ij}.

[6] Butlin's argument is directed at GPT-3 which, at the pre-training stage at least, is trained to predict the following word (check) and sentence. At the fine-tuning stage matters are more complex and vary between tasks. One may wish to characterise everything achieved with a soft-max layer as 'prediction' but this is an interpretation. While the argument in this section is generally opposed to Butlin's characterisation, the conclusions will be rather similar.

This argument, in general, has three steps:
(1) The intentions of designers determine the function of an artefact.
(2) The intended function of a neural network is the network's objective function.
(3) The objective function of a language model is to predict conditional probabilities between words.

I want to dispute both that the intentionalist approach is appropriate when the tools under consideration are neural networks (1), that designers actually intend a network to merely compute its objective function (2), and the claim that this limits us to word prediction. To do this, we need to look at the idea of an artefact function and ask whether it might need to be modified when the kinds of artefacts we are considering undergo training.

The concept of an 'artefact function' plays its primary role in the explanation of an artefact's structure and to explain instances of malfunction. A can-opener has the structure it does because that enables it to fulfil its function and it malfunctions when it no longer opens cans, a cup malfunctions when it leaks etc. There are several problems with the idea that designer intentions determine the artefact functions of neural networks. Here, I will discuss the division of labour in engineering and the fact that designers' intentions often overstep their capabilities.

The division of labour within engineering ensures that any complex computational system will have multiple designers with no guarantee of overlapping intentions. Lines of code developed with one intention may be reappropriated (e.g. copied and pasted from Stack Exchange) to serve another. In these cases, it's not clear whose intentions should matter in our explanations. The second, is that the avowed intentions of designers often go beyond anything found in the objective function. In the case of Word2Vec, the stated intention in Mikolov et al. is to produce vectors representing the *similarity* of words to each other.[7] While the objective function was to compute the conditional probability that words would occur in the same context, the model was *assessed* on a range of word similarity tasks (Mikolov et. al., 2014). If the designer's intention was sufficient to fix the proper function of an artefact, we would have to concede that this is what the network functions to represent (it might always fail but the question of function would be decided). Similarly, if the production of PCA maps, were why researchers developed and trained word2vec, then the function of word2vec would be to produce PCA world maps. As a result, appeal to intentions, if truly constitutive of function, might suggest that a system like GPT-3 engages in conversation about the real world (if that was the designer's intention). This is a particularly serious issue when the intentions of researchers in AI may be far more ambitious than their technology could realise. On the other hand, we also want to be able to say that designers have failed to realise their intentions when designing a network, for example, it may be that a given network doesn't represent crime rates but house prices, even when designers intended it to represent the former. For these reasons, I think we should avoid grounding our notion of artefact function in designer intention, at least in the case of neural networks.

Here's a rough story about how design-intention and function might interact. Consider a researcher who wants to classify handwriting (perhaps they are a first-year machine learning student). To realise this intention, they code a small neural network and go about setting its various hyperparameters. They also select a data set (most likely the MNIST database) and separate it into training data and test data. They choose the number of layers, functions computed at each layer, batch size, dropout rate etc. Part of this project involves selecting an objective function and for this, again, they have several options (e.g., categorial cross-entropy, large margin cosine loss, triplet loss). It is by making a range of choices like this that the researcher comes to realise their intention through the device. But they can only fix the (design) function of that over

[7] "The quality of these representations is measured in a word similarity task" (Mikolov et al, 2013).

which they have control. This brings us to the central point. The researcher has control over the network's *hyperparameters*, but the network's *parameters*, the actual weights and balances, are set during training. It is ultimately the configuration of the parameters which come to represent things. Unlike can-opener designers, computer scientists and engineers are not in a position to specify the structure of a language model (if they were, the field of machine learning would be somewhat redundant). The conflation of neural networks and standard artefacts depends upon this conflation of hyperparameters and parameters.

————

The completed version of this paper might have a section here which discusses the complex interactions between designer's intentions and networks' representational capabilities. I'm trying to develop an adequate theory for what I call 'stochastic measuring devices', a term I have coined that I take to be much more sensible and descriptively adequate than 'artificial intelligences' but I suspect this is actually a task for a different paper.

————

4.2 Applying the Framework

Returning to the Word2Vec algorithm, a teleosemantic analysis needs to identify mechanisms for producing and consuming representations, units of selection, and units of representation. It will also be helpful to distinguish embeddings-in-training from trained-embeddings. The notion of representation is clearer in training as the embeddings are playing a direct role in assisting the model to perform a task, but it still makes sense to say that trained-embeddings can represent what they were trained to represent, as the orchid Ophrys Apifera still mimics (i.e. represents) a now extinct bee. While, due to the extinction of the bee, the orchid leaves no longer have directive representational content, they still have descriptive content.

The mechanisms which evolve during training to produce and consumer representations are the matrices, M and M′ and they each evolve to serve both functions. Matrix M evolves to consume encoding vectors and in response produce representations (i.e., embeddings in the hidden layer) which are then consumed by matrix M′ as it performs its task. M′ evolves to consume embedding vectors and to respond by producing output vectors. These matrices meet Millikan's criteria for a selection process. Each generation bears similarity to the previous generation (in a sense to be elaborated), changes according to a natural law (i.e., the update function, $v_{tn+1} = v_{tn} - \eta \cdot e \cdot h$), and depends counterfactually on their previous state.[8] However, unlike what happens in traditional accounts of selection, the training process does not 'preserve' values and pass them on to the next generation. Each node and weight trivially survive, (even if set to zero). Instead, the values which contribute to the representational significance of the network are altered in accordance with the update function. This means that we need to rethink what we mean by 'selection' for this context. The natural place to make the change is with the 'similarity' criterion. While the values at v_{tn+1} and v_{tn} are going to be 'similar', they need also to be different for training to occur. Let's then say that selection occurs if and only iff the loss function is minimised by alteration of the matrix values. In a sense, the steeper the gradient at a value, the more that value is selected.

Next, we need to account for the vehicles of representation. As indicated, I take the vectors produced by the matrices to be the relevant vehicles. This means that there are actually three sets of representations in the network, the encoding layer, the embedding layer, and the output layer. It's worth distinguishing this from alternative

8 Superficially, a stochastic process shouldn't result in a counterfactual dependence between states, however, the sense in which stochastic gradient descent is actually stochastic is much weaker. It is stochastic because the descent algorithm works with samples taken from the data set rather than using the whole data set at once. Any local applications of the update function are wholly deterministic, thus satisfying the counterfactual dependence condition.

approaches. In early work on connectionist systems, Andy Clark argued that the unit of representation should be the individual node (Clark, 2001) while Jerry Fodor argued that the smallest unit of representation for a connectionist system was the whole network (Fodor, 2000). The appeal of both of these approaches is clear. When we look at the vector space representation of a language model, it appears that different dimensions should come to represent different microfeatures of the words modelled. At the same time, holism also appears appropriate since the vector space has no endogenous interpretation. The centre of the vector space doesn't come with a pre-determined meaning. Contrast this with the standard three-dimensional colour space model with dimensions representing hue, brightness and saturation (see Lee, 2021). The geometric space of the colour space model has its own interpretation, but the vector space of word-embeddings does not.[9][10] In response to Fodor's holist interpretation, the claim that individual vectors have no representational content independent of the other vectors in the system, we should distinguish *geometric holism* from *functional holism.* In the case of geometric holism, the significance of the vector is determined by its relation to all the other vectors in the semantic vector space. This holism requires a measure of distance between representations (e.g., cosine similarity or dot-product values) which in turn determines the meaning of the space of representations. The rhetoric of geometric holism is prevalent in structuralist and post-structuralist thought. In the case of functional holism, a representation constitutively performs multiple functions where the set of functions the representation can perform determines or constrains the directive content of the representation. A set of functionally holistic representations can be given a geometric interpretation but this interpretation is not what determines content.

Word2Vec embeddings are not geometrically holist as they don't use the similarity measure or geometric operations to determine outputs. Instead, the values of the vectors are fed into the output mechanism to select across the rows in the M′ matrix. The content of an embedding vector is determined by how the distribution it produces after being 'consumed' by M′ (multiplied and softmaxed) serves to minimise the output of the loss function over time, it is not determined by the distance between the vector and every other vector in the vector space. This is still holist, the content of a vector is determined by how that vector relates to every other word in the vocabulary via the probability distribution it produces over those words, but it is not determined by its relation to every other vector in the vector space.

[Reference literature on 'distributed representations' in connectionism - e.g. , Godfrey-Smith 2009]

As we described above, to be a representation is to be 'taken to mean something' by some consumer system. The matrix M′ takes the word embedding to mean something by using it to produce a probability distribution over the words in the vocabulary. This interpretation of the matrix's behaviour relies upon the fact that the loss function has historically punished it for failing to do this job successfully. At any point in the past training when the output of M′ diverged from the appropriate distribution of words, the backpropogation algorithm ensured that the weights which constitute M′ were changed with the exact rate of change corresponding to how far the output generated was from

[9] This does not mean that the structure is arbitrary. In fact, there is some evidence that across different languages 'good' vector spaces (i.e. those produced with sufficient data) are isomorphic (Vulić et al., 2020).

[10] Shea argues that the appropriate unit of representation, at least for some connectionist systems, is a cluster within the possible state space of the network. While this may be appropriate for systems implementing non-linear functions (e.g., tanh, ReLU or sigmoid) it isn't applicable to word2vec. The reason non-linearity matters is that it means that slight changes to the values themselves, up to the edge of the cluster, may not ultimately alter output values. There is a degree of tolerance in virtue of the non-linearity of layer transformations. In contrast, the layers of word2vec, with the exception of the output layer, compute linear functions in which the change in input is met with a linear change in output. This motivates the following conjecture: non-linear relations between layers may be best interpreted as transformations on the format of information while linear-transformations may be best thought of in producer-consumer terms, performing task functions which themselves contribute to the *content* of information.

the actual distribution found in the environment of the training data. The ability to approximate the distribution of words in the training data is what leads to a minimisation of the output of the loss function and so it is this which fixes the content of the embeddings.

4.3. Results of Applying the Framework

At the start of this paper, we set two conditions on a theory of content for a system, it must explain the role of our content ascriptions and it constrain ascriptions of content. So how does teleosemantics do here? According to the teleosemantic framework, neural word-embeddings are a kind of intentional sign. They are produced by a producer-system which has been designed by an evolutionary process to output states serving further downstream tasks. Neural language models exhibit a primitive form of (original) intentionality. This immediately distinguishes them from traditional forms of computation which appear to possess purely derived intentionality.

As mentioned, teleosemantics doesn't provide a simple algorithm method for determining the representational contents of a system, but it can provide us with some constraints. The network we are looking at has three layers, each of which functions as a PPR, that is, each layer has both descriptive and directive content. We'll look at the two outer layers first. The encoding layer has the simplest kind of content. This layer 'represents' an individual word in a weak, Dretskean sense — a particular one-hot encoding is reliably tokened for each word type in the dataset. In being tokened, the vector classifies a given token in the training data as a token of a predetermined type, in other words, it represents that token as an instance of a type.[11] The descriptive content of the vector is the token in the training data and the directive content is to instruct M to produce an embedding vector. Unlike the vectors downstream from it, the system did not evolve a producer system to generate this vector, it was not learned, but the mechanism which consumes it was a product of evolution.

On the other side of the system, the output vectors are produced by M' and consumed by the backpropogation algorithm. The states of affairs in the environment which they are produced to approximate are the particular distributions with which they are contrasted during backpropogation and which the error function is determined against. This is because the proper function of matrix M' is to produce distributions which better approximate those found in the training data. A particular output vector's representational fidelity is assessed relative to a particular distribution and the mechanism M' has been modified in response to representational failures. During any given run, then, we can say that this vector represents (to a variable degree of accuracy) the distribution against which it is assessed. It is because of these processes of assessment, the computation of error by the backpropogation algorithm, that we can interpret the softmax function as computing a probability distribution and say that the output vector represents a probability distribution over the words in the data set.

Finally, we can look at the embedding layer. Again, we find that teleosemantics can make a useful contribution. For example, we can rule out the possibility that word-embeddings represent their most proximal stimulus. The most proximal content-ascription we could make for the language model would be that the word-embedding vector represents the encoding vector which causally precedes it. In other words, an embedding vector [3, 4, 2, 1, 6, 7…] represents a one-hot encoding vector [0, 0, 0, 1, 0….]. We can rule this out because the one-hot vector would not feature in a Normal explanation of the function of the producer mechanism. This is because the encoding vector doesn't occur in the dataset. Just as the concept CAT does not represent the nerve-stimuli which might token it and edge-detector cells do not represent light intensity gradients on the retina (they detect edges), a word-embedding does not represent the vector which encodes it. A neural word embedding, like Dretske's

[11] There are a variety of ontological issues connected to this concerning the metaphysics of digital words and the metaphysics of words more generally.

percepts, "skips over (or 'sees through') the intermediate links in the causal chain in order to represent . . . its more distant causal antecedents" (1981, p. 158).[12]

The directive content of the embedding layer is to instruct the matrix M' to produce a probability distribution. But should we say that this distribution is also the descriptive content of the representation? Or should we perhaps say that the embedding represents the same token word which triggered the encoding vector? Again, let's consider how M and M' have evolved during training. During training the vectors produced by M' are contrasted directly against particular distributions in the training data (where 'contrasted' means that cross-entropy is computed). This is not the case for the embedding vectors produced by M. They are not produced to be directly contrasted with any *particular* distribution, rather they are produced to produce output vectors which can be used in this way. Unlike the encoding vectors which do not change throughout training, the embedding vectors are produced by a mechanism whose evolution has involved exposure to any number of inputs in the data set.

In light of these considerations, I think it is most appropriate to say that the embedding vectors represent the distribution of a word across the data set. The mechanism M produces a representation whose function it is to input into M' and as a result produce a probability distribution. *The directive content of this representation is the probability distribution it, by interacting with M', produces (i.e., the output vector). The directive content is what M' is supposed to produce in response to the embedding. The descriptive content is the actual probability distribution in the data set which the output vector is assessed against. It is the part of the world which the embedding's producer has evolved to produce representations of.* When training is complete the set of word embeddings which the network has produced no longer have directive contents and so no longer represent the set of output vectors. But they still have the descriptive content which characterises the state of the affairs which gave rise to them. In this case, the distributional relations between words in the data set. This means that word-embeddings do not represent semantic similarity relations, the geographic locations of cities, or the size of animals. The theory constrains the contents we might ascribe to the language model.

[According to Millikan, declarative sentences are descriptive because their function is to produce true beliefs in their consumers (i.e. hearers). They don't represent the world independent from how they are consumed. The true beliefs they produce are all representations. Their directive function is to produce representations and the descriptive content is given by the success conditions of those (directive) representations.]

Similar vectors produce similar distributions but they do not produce the distributions they do in virtue of calculating similarity relations with each other. Any isomorphism between the similarity relations induced by word embeddings and possible 'objective' similarity relation between 'words', 'word use' or something else, goes unexploited by the word2vec algorithm. Neither does the algorithm perform principle component analysis in the course of its run-time (though it does perform dimensionality reduction). A consumer-based teleosemantic theory ascribes word-embeddings very weak content. One might find this disappointing but it does conform with alternative conditions on representation. Consider, for example, whether we should take a PCA map to provide a cartographic representation. If we did so, we might ask what would happen if the cities of Europe had different locations but the text in the data set remained the same. In this case, the network would have come out with the same results as it would not be counterfactually sensitive to any geographic information

[12] I have not been able to find any studies indicating that one can recover one-hot encoding from word-embeddings but this seems like a worthwhile experiment. If encoding vectors cannot be recovered, this would provide further evidence than embeddings are intentional representations and not natural signs (Millikan, 2002: 81-82).

beyond the relations in the data set. The word vectors are only counterfactually sensitive to text in the training data and it does not perform this task any better or worse if the sentences in the dataset are true or false.[13]

The framework can also shed light on the significance of attention mechanisms in Large Language Models. The core idea behind attention heads (in transformer models but also seq2seq neural translation models) is to exploit the geometric relations between vectors in order to produce better representations. To use the paradigm case, scaled dot-product attention utilises the dot product between two embedding vectors to determine the extent to which one embedding should 'attend' to another during training (Vaswani et al. 2017). It is the exploitation of previously exogenous representational content which gives transformers their power. In contrast to Butlin then, GTP systems, as exploiters of the geometric information contained in word embeddings, may represent the similarity relations between words, and even syntactic relations (Lappin, 2021), despite the fact that the final layer performs, during pre-training at least, a standard language modelling task. There is reason to think that attention heads enable deep neural networks to move beyond simple pushmi-pullyu signs and perform operations which involve reasoning/calculating over signs. A consequence of this is that two identical embeddings may represent different contents depending on the network using them. If that network has attention heads, then the geometric distance between vectors is exploitable and, assuming that the exploitation of that information informs training, it is appropriate to say that the distance between vectors is represented at that layer. Whether the representation is exploited would be determined as above by probing the gradient of the loss function at the attention layer.[14]

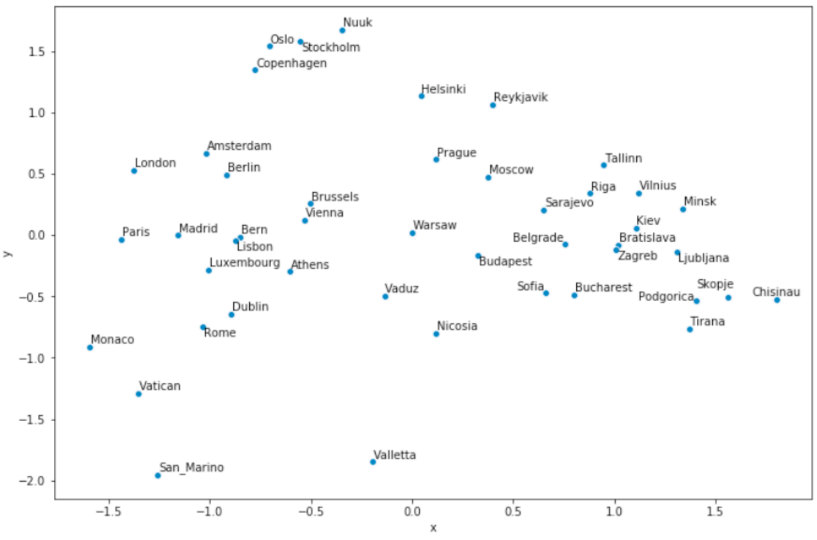
Conclusion

In this paper, I have presented an application of the teleosemantic framework to one particular language modelling algorithm. The purpose of this has largely been as a proof of concept, an attempt to demonstrate the possible value of teleosemantics in this field. The most important results here are negative, in particular, the claims that word embeddings don't represent cartographic information or even 'semantic similarity' relations. However, this second claim should be viewed as more tentative than the first. What matters is that the method does provide a criterion for distinguishing genuine representational contents within the system from those imputed to the system by its users. Despite this small success, we are still left with many further questions. Can the framework shed light on the difference between one- and few-shot learning? Can it distinguish what is unique about deep contextualised embeddings (e.g. ELMo) and how they relate to claims about word 'senses'? Can it give any further insight into the representational properties of attention heads? How does it relate to particular probing methods (Søgaard, 2021) and, importantly, does it call any existing probing methods into question? Can the 'distributional hypothesis' be precisified to the point that it can bridge claims about probability distributions and semantics?

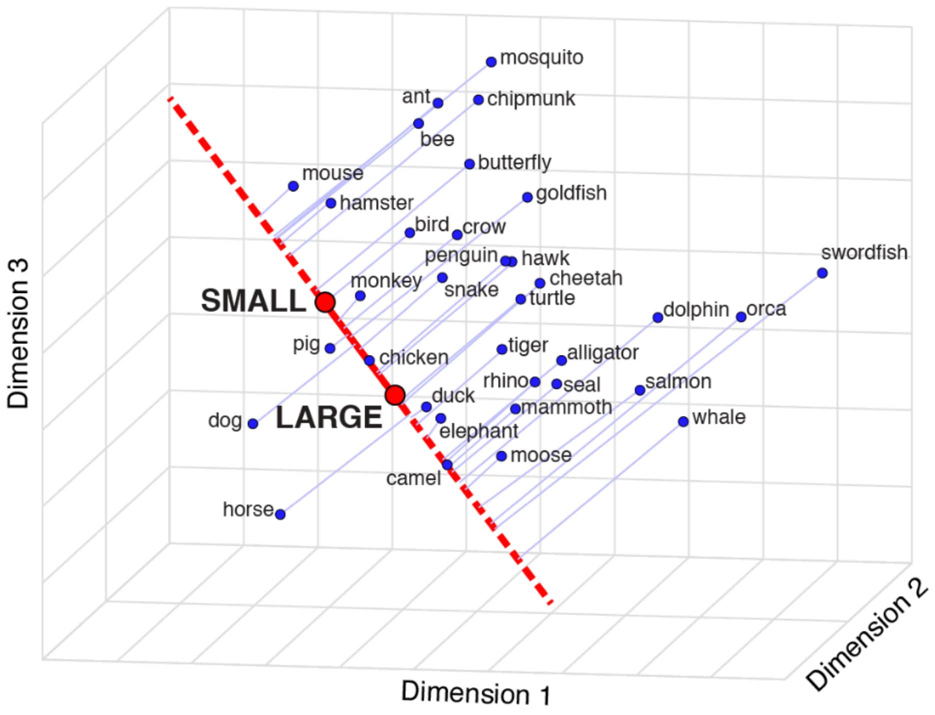
[13] It is a major assumption to interpret the dataset as the environment and not as part of the system. If we were considering the UNS, we might view the dataset as part of the system which is sensitive to the world. How this relationship should be thought about is an exceptionally complicated matter which is being set aside for the purposes of this paper.

[14] While I don't know of any network layers which explicitly perform PCA, some layers can be best understood as performing dimensionality reduction of other kinds. Word2Vec itself reduces the number of dimensions of a one-hot encoded vector to the dimension size of the word-embeddings.

Appendix 1: Graphs



Taken from: https://towardsdatascience.com/how-to-draw-a-map-using-python-and-word2vec-e9627b4eae34



Graph of semantic subspace projection on a small-large scale, from Grand et al. 2022

Appendix 2: Hierarchical Softmax and Fine-tuning

The next question we need to address concerns fine-tuning. There are several ways of modifying word2vec to produce better results, most obviously, by replacing the softmax function with a hierarchical softmax function or by supplementing the regular training objective with negative sampling. The question we need to ask in these cases is whether the modified systems represent the same thing as the unmodified system, or whether they represent something else? Do they represent something *better* or do they represent something *different*? While purely internalist accounts of representation have historically struggled to answer this kind of normative question, teleosemantics is better placed to shed some light. To answer the question we need to ask, what was the relation between the system's response (i.e., the final layer) and the environment (i.e., data set) in virtue of which the loss function was minimised. Recall that the conventional softmax function produced a probability distribution from a set of output scores. In other words, the network had to produce an output score for every possible word type in the data set. With hierarchical softmax, the set of possible outputs is hierarchically structured as a branching tree with each output represented as leaves. Instead of requiring that the network compute an output value for every possible output, what the network needs to do is predict the correct path along the branches of the tree to the relevant leaf. This cuts the number of predictions down from |V|-many to something like $\log_2|V|$ predictions, significantly cutting computational cost. Nevertheless, while the network is no longer directly computing a probability distribution for each possible output words, that is, it is no longer the case that, for each word, a distinct probability is computed based on the previous state of the system, it remains the case that the system is indirectly producing a probability distribution all words. With negative sampling, the network is shown word-context pairs which either do or don't occur in the data set (the negative samples are randomly generated based on the dataset). The loss function measures the distance between the network's prediction and the right prediction which is either 1 or 0. In this case, again, the loss is minimised by the network's predictions better approximating the actual relations in the data. As a result, we can still say that these modified versions of the algorithm represent *the same thing* (the conditional probabilities distributions of co-occurring words) as the original algorithm even though they use a slightly different method. However, it is unlikely that these answers would be the same for generative pre-trained models (e.g. BERT, RoBERTa, GPT-2) which are trained upon multiple tasks and the application of teleosemantics to this framework must be left to future research.

Bibliography

Buckner C. 2021 . A Forward-Looking Theory of Content. *Ergo.*

Butlin, P., 2021. Sharing Our Concepts with Machines. *Erkenntnis*, pp.1-17.

Clark, A. 2001: *Mindware.* Oxford: O.U.P.

De Cosmo, 2022, Google Engineer Claims AI Chatbot Is Sentient: Why That Matters, https://www.scientificamerican.com/article/google-engineer-claims-ai-chatbot-is-sentient-why-that-matters/

Fodor, J. 2000: The Mind Doesn't Work That Way Cambridge, MA, MIT Press.

Grand, G., Blank, I.A., Pereira, F. *et al.* Semantic projection recovers rich human knowledge of multiple object features from word embeddings. *Nat Hum Behav* **6**, 975–987 (2022). https://doi.org/10.1038/s41562-022-01316-8

Ivanova, A., Hewitt, J. & Zaslavsky, N. 2021. Probing Artificial Neural Networks: Insights from Neuroscience. ICLR 2021 Workshop "How Can Findings About The Brain Improve AI Systems?"

Lappin, S. 2021 Deep learning and linguistic representation. CRC Press

Lee, A Y. 2021. Modeling Mental Qualities. _The Philosophical Review_ 130 (2):263-209.

Linzen, T. & Baroni, M. 2021 Syntactic Structure from Deep Learning. Annual Review of Linguistics. Vol. 7:195-212

Mallory, F. 2020 Linguistic types are capacity-individuated action-types, *Inquiry*, 63:9-10, 1123-1148

Mallory, F. 2023. What do large language models model? In Communicating with AI (eds. Herman Cappelen & Rachel Sterken)

Mallory, F. 2023 Wittgensteinian Language Modelling, *Manuscript*

Mikolov, T., Chen, K., Corrado, G. and Dean, J., 2013. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. and Dean, J., 2013. Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems, 26.

Millikan, R G. 1984. *Language, Thought and Other Biological Categories. Cambridge, MA: MIT Press.*

Millikan, R G. 1989. 'Biosemantics', Journal of Philosophy, 86: 281–97.

Millikan, R G. 1990. 'Truth Rules, Hoverflies, and the Kripke-Wittgenstein Paradox', Philosophical Review, 99: 323–53.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. 2018. Deep contextualized word representations. In Proceedings of NAACL, New Orleans, LA, USA.

Piccinini G. Situated Neural Representations: Solving the Problems of Content. Front Neurorobot. 2022 Apr 14;16:846979. doi: 10.3389/fnbot.2022.846979. PMID: 35496901; PMCID: PMC9049929.

Rong, X., 2014. word2vec parameter learning explained. arXiv preprint arXiv:1411.2738.

Shea, N. 2007a. 'Content and Its Vehicles in Connectionist Systems', Mind & Language, 22: 246–69.

Shea, N. 2014a. 'Exploited Isomorphism and Structural Representation', Proceedings of the Aristotelian Society, 64: 123–44.

Shea, N. 2018. Representation in Cognitive Science. Oxford University Press.

Søgaard, A. 2021. Explainable Natural Language Processing. Synthesis Lectures on Human Language Technologies. 14(3) pp. 1-123

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., 2017. Attention is all you need. Advances in neural information processing systems, 30.

Vulić, I., Ruder, S. and Søgaard, A., 2020. Are all good word vector spaces isomorphic? Proc. of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 3178–3192, Online. Association for Computational Linguistics. DOI: 10.18653/v1/2020.emnlp-main.257.