Categorical Equivalence of Word Embeddings

Ares Fabregat-Hernández[1] arfabher@etsii.upv.es
Javier Palanca[1] jpalanca@dsic.upv.es
Vicent Botti[1,2] vbotti@dsic.upv.es

[1] Valencian Research Institute for Artificial Intelligence (VRAIN)
Universitat Politècnica de València, Camí de Vera s/n, 46022, Valencia, Spain
[2] valgrAI (Valencian Graduate School and Research Network of Artificial Intelligence)

Explainable Artificial Intelligence (XAI) is a growing field that aims to increase the transparency and interpretability of machine learning models. The aim of this work is to use the categorical properties of learning algorithms in conjunction with the categorical perspective of the information in text datasets to give a framework to explain word embeddings. More precisely, we use the structure on the syntax category of a text to compute distances and probabilities. With this information we prove that word embeddings achieved via metric multidimensional scaling (mMDS) satisfy a stability condition: Nash's equilibrium. Furthermore, we show that the embeddings obtained via the CBOW and Skip-gram algorithms are equivalent to the ones obtained via mMDS thus having the same stability properties. As a consequence, we deduce the linear/logical properties of the embeddings. We end the manuscript by emphasizing the generality and applicability of the approach.

1 Introduction

Explainable Artificial Intelligence (XAI) is an increasingly important field of research aimed at developing AI systems that can be transparently understood by humans. The concept of XAI is motivated by the need to address the issue of the so-called "black box" problem in AI, where machine learning models make predictions or decisions that are difficult to interpret by humans. This lack of transparency can be a significant obstacle to the adoption of AI in various fields, including healthcare, finance, and legal domains, where the consequences of AI decisions can have a profound impact on human lives (see (9)).

Natural Language Processing (NLP) has become an increasingly important field in recent years, with many applications ranging from chatbots to sentiment analysis. However, much of the recent progress in NLP has been achieved using complex deep learning models such as convolutional and recurrent neural networks. These black-box models, although effective, are often difficult to interpret, and it can be challenging to understand how they make their predictions.

The lack of interpretability of these models is a significant concern for many NLP applications, particularly those that require high levels of accuracy and robustness, such as legal or medical document analysis. In many cases, it is not enough to know whether a model's output is correct, but also to understand why it produced that output.

We propose to attack these problems using category theory. Indeed, category theory is a branch of mathematics that deals with abstract structures and relationships between them. It provides a powerful framework for understanding the behavior of complex systems, including machine learning models.

Submitted to:
ACT 2023

Category theory can be applied to machine learning models in multiple ways. Firstly, it can describe the universal properties of mathematical objects, such as neural networks, as functors between categories, which can formalize the concept of "learning" as a natural transformation. Secondly, category theory emphasizes compositionality, which is important in machine learning as it involves building complex models from simpler building blocks, such as layers in a neural network. Category theory can provide a rigorous framework for understanding how these building blocks can be combined and how resulting structures behave. Lastly, topos theory, a branch of category theory, can be used to study the structure of spaces and their relationships, including data sets in machine learning.

It is in this context that Applied Category Theory can help. By studying the structure of text as in (1) we can deduce patterns that get picked up by the algorithm. Furthermore, by incorporating the category theoric structure on the algorithms themselves, as in (2) we can keep track of how those patterns propagate from input to output.

2 Theoretical background

In this section we discuss the machine learning prerequisites for the rest of the paper. The first part is devoted to the n-gram probability distributions and the second part is devoted to the enriched concept of learners.

2.1 n-Gram Probability Distributions

In order to understand natural language models via category theory we need first to understand how they work computationally. Let T be a text, and $w \in T$ a word in the text. If we want to know the probability that the next word in a sequence h of T is w we can estimate the conditional probability $P(w|h)$ by counting the relative frequencies:

$$P(w|h) = \frac{C(hw)}{C(h)},$$

that is, the number of times that the sequence $t = hw$ appears in T divided by the number of times the sequence h appears in T. If t is a sequence of n words we will also write $t_{1:n}$ to emphasize its length. Now, using the chain rule of probability we can reduce the problem of computing the joint probability $P(t_{1:n})$ to computing the product $\prod_{k=1}^{n} P(t_w|t_{1:k-1})$ of conditional probabilities. The problem with this approach is that if the length is too big, a particular combination of words might not have occurred in a particular text T. That is why we approximate the result by working "locally": by only considering the probability of extensions of k of words

$$P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-k+1:n-1}) = \frac{C(w_{n-k+1:n})}{C(w_{n-k+1:n-1})}.$$

2.2 Enriched Category of Learners

The datasets are not only set in the mathematical sense: they carry more information. This information can be seen as a metric of similarity between data points. If we consider the data sets as metric spaces we can consider the enriched version of the framework proposed in (2). This means, **learners** are elements of a category whose objects are metric spaces and whose morphisms of the form (P,I,U,r) where P is a set and

$$I\colon P\times A\to B \quad U\colon P\times A\times B\to P \quad r\colon P\times A\times B\to A$$

are functions such that for any $p\in P$ the implement function $I(p,-)\colon A\to B$ satisfies the Lipschitz condition. In particular, the category of learners can be produced from the category of parameterized function (see (2, Theorem III.2)). Since for all $p\in P$, the function $I(p,-)$ is Lipschitz, we can associate to every morphism in of parameterized functions $(P,I)\colon A\to B$, a parameterized function $(P,L_I)\colon \{*\}\to\mathbb{R}_+$, where $\{*\}$ is a one-point space and $\mathbb{R}_+=\{x\in\mathbb{R}\colon x\geq 0\}$. That is, $L_I\colon P\times\{*\}\to\mathbb{R}$ sending $p\mapsto L_I(p,*)=L$, where L is the Lipschitz constant of the function $I(p,-)$. Hence, we can enrich the category of parameterized functions over the set of parameterized functions $(P,L_I)\colon\{*\}\to\mathbb{R}_+$, which we denote by $\mathcal{L}$, up to equivalence of parameterized functions. This set is actually a monoid with the monoidal operation given by $(P,L_1)\circ(Q,L_2)=(P\times Q,L_1\cdot L_2)$ where the product of functions is the pointwise product. The monoidal neutral element is the parameterized function $(\{*\},e)$ with $e(*,*)=1$. That is, given two parameterized functions $(P,I)\colon X\to Y$ and $(Q,J)\colon Y\to Z$ the product function is

$$(P\times Q,L_I\cdot L_J)(p,q,*)=L_I(p,*)\cdot L_I(q,*)\in\mathbb{R}_+$$

for all $(p,q)\in P\times Q$. Thus, $(\boldsymbol{\mathcal{L}},\circ)$ is a monoid. We can then turn $(\boldsymbol{\mathcal{L}},\circ)$ into a monoidal category, also denoted $\mathcal{L}$, by letting $(P,L_I)\colon\{*\}\to\mathbb{R}_+$ be its objects. The morphisms are maps from one parameter space to the other $f\colon P\to Q$ and composition is given by the composition of maps between sets. The monoidal product is $\circ$ and the monoidal identity is $(\{*\},e)$. Most machine learning algorithms fit in this category and, in particular, neural networks can be understood as a subcategory of the category of learners. This means we can track and bound the distortion the algorithm is going to apply to the metric of the data set (see (5)).

3 Categorical Modeling

In this section, we discuss the categorical notions necessary to extract information from a corpus of text and have a mental picture of the situation. For this, we first study the space attached to a text as a topological space which gives us the notion of closeness via neighborhoods. This point of view has a major limitation: we cannot compare two words directly. In order to have an adequate framework to compare words we will introduce a categorification of the topological space where, via the Yondeda embedding, we will have a way to give a distance between words.

Let T be a text. We can define the space X_T of n-grams of the text. By that, we mean the space of finite sequences g that appear in the text T. We define the **length of g,** $l(g)$ as the number of words in contains. This space can be endowed with the Alexandrov topology. Indeed, we can

define a preorder in X_T via containment: $g \leq t$ if and only if g is a subtext of t or, equivalently if t is a context of g. This means that for every $g \in X_T$ we have a smallest neighbourhood:

$$U_g = \{t \in X_T \colon g \leq t\},$$

that is the set of all sequences in T containing g. Conceptually, this is the set of all sequences of T that contain g thus making it the set to take into account to compute conditional probabilities of extensions. We can refine those sets by filtering by the length of the sequences:

$$U_g^n = \{t \in X_T \colon g \leq t,\ l(t) \leq n\}.$$

In this case, we are only looking at sequences of length at most n that contain g. Finally, we can grade the space X_T using the lengths of sequences:

$$X_T = \bigoplus_{n \geq 1} X_T^n \tag{1}$$

where X_T^n is the set of sequences of length n. In this way, the set X_T^1 is the set of words that appear in T, also called the dictionary of T, the set X_T^2 is the set of bigrams of T and so on. Thus, we can rewrite the sets U_g^n as the n-th filtered piece of U_g

$$U_g^n = \mathcal{F}_n(U_g) = \bigoplus_{m \leq n} (U_g \cap X_t^m).$$

One thing to note is that when solving this problem computationally we use padding by <EOS> symbols. This has the effect of always having a sequence of length n around a word, even if it is the first word of T. We carry this assumption to the following lemma.

Lemma 3.1. *Let T be a text and X_T its associated Alexandrov space. Then, for all $g \leq t$, we have*

$$P_n(g|t) = \frac{1 + \sum_{k=1}^{m} 2^{k-1} C(t)}{1 + \sum_{k=1}^{m'} 2^{k-1} C(g)} \frac{C(t)}{C(g)} = \varepsilon_n(C(t), C(g)) P(g|t-g) \tag{2}$$

whith $\varepsilon_n(C(t), C(g)) \leq 1$.

Proof. First of all, notice that for all $x \in X_T$ we have $C(x) = |U_x^{l(x)}|$. Furthermore,

$$\begin{aligned} |U_x^{l(x)+1}| &= C(x) + |\{s \in X_T^{l(x)+1} \colon x \leq s\}| \\ &= C(x) + |\{s \in X_T^{l(x)+1} \colon s = tw \text{ or } s = wt \text{ with } w \in X_T^1\}| \\ &= C(x) + 2C(x). \end{aligned}$$

Let $m = n - l(t)$ and $m' = n - l(g)$. Then, the quotient can be written as

$$\begin{aligned} P_n(g|t) = \frac{|U_t^n|}{|U_g^n|} &= \frac{\sum_{k=0}^{m} 2^k C(t)}{\sum_{k=0}^{m'} 2^k C(g)} = \frac{1/C(t) + \sum_{k=1}^{m} 2^{k-1} C(t)}{1/C(g) + \sum_{k=1}^{m'} 2^{k-1} C(g)} \frac{C(t)}{C(g)} \\ &= \varepsilon(C(t), C(g)) P(g|t-g). \end{aligned}$$

Notice that since $g \leq t$, $m' \geq m$ and $C(g) \geq C(t)$. Thus $\varepsilon_n(C(t), C(g)) \leq 1$. □

The last part of the proof implies that the conditional probabilities obtained with the Alexandrov topology on X_T coincide with the conditional probabilities obtained by counting occurrences on T. Thus, by analyzing the topological properties of the space X_T we can get the same information as the n-gram probability models.

The problem is that this picture is insufficient to generate word embeddings, which are crucial for NLP tasks since we cannot compare the distance between two words. Indeed, to have such a comparison we need one element to be in a higher graded piece whereas two words belong to the same graded piece X_T^1. To solve this problem we need to reinterpret the space X_T as a category. Since we have a preorder on X_T we define $\mathcal{C}_T$ as the category whose objects are sequences of T. Given two objects x, y of $\mathcal{C}_T$, there is a morphism $x \to y$ if and only if $x \leq y$. The category $\mathcal{C}_T$ shares many interesting properties. We recall the definition of a **graded category**

Definition 3.2. *Let $(M, \otimes, I)$ be a monoidal category. Then for a given category $\mathcal{C}$, an M-**graded monad** is a lax monoidal functor from M to the endo-functor category $\mathcal{C}$*

$$(M, \otimes, I) \to ([\mathcal{C}, \mathcal{C}], \circ, id_{\mathcal{C}}).$$

In particular, the grading may arise from a monoid $(M, \otimes, e)$. The for a category $\mathcal{C}$ we would have a family of endofunctors T_m indexed by the elements of M with maps

$$\mu_{m,n} \colon T_m(T_n(X)) \to T_{m \otimes n}(X)$$

and $\eta_X \colon X \to T_e X$ with X and object in $\mathcal{C}$. If the monoid is $(\mathbb{N}, \times, 1)$ then the functors T_n give the length n elements. We want to adapt this definition to have the set of n-grams of $\mathcal{C}_T$.

Definition 3.3. *Let T be a text and $\mathcal{C}_T$ the category of sequences of T and let $(\mathbb{N}, \times, 1)$ be the monoidal category whose objects are positive natural numbers, there is an arrow $m \to n$ if $m \leq n$ and the monoidal product is defined by multiplication. We define the n-th graded piece of $\mathcal{C}_T$ as the image of $\mathcal{C}_T$ by $T_n \in ([\mathcal{C}, \mathcal{C}], \circ, id_{\mathcal{C}})$. We denote it by $\mathcal{C}_T^n$.*

One thing we gain with the categorical perspective is access to the Yoneda embedding. Let $x \in \mathcal{C}_T$, we denote by $y^x = \mathcal{C}_T(x, -)$ the Hom-object represented by x. This will help us keep track of all the information relative to x. For instance, we can describe the sets U_g^n using the restriction of y^g to the n-th graded part of $\mathcal{C}_T$:

$$|U_g^n| = |\operatorname{Im}(y^g_{|\mathcal{C}_T^n})| \tag{3}$$

thus yielding an alternative proof of Lemma 3.1. We can summarize the algorithm to get a n-gram probability model as follows:

Computer Science: We parse the text T and count instances of $w_{1:n}$ and $w_{1:n-1}$ and then compute the quotient.

Topological Space: We compute the cardinalities of the smallest neighborhoods $|U_{w_{1:n}}|$ and $|U_{w_{1:n-1}}|$ and compute the quotient.

Category Theory: We compute the images of the functors represented by $w_{1:n}$ and $w_{1:n-1}$ and compute the quotient.

The last perspective indicates that given an arrow $x \to y$ we can attach to it the probability $\mathcal{C}_T(x,y) = P(y|x)$. This means we can enrich the category $\mathcal{C}_T$ over the monoidal preorder $([0,1],\leq,\otimes,1)$ as in (1). Furthermore, by composing the conditional probabilities with $-\log$ we get a number in $\mathbb{R}_+ = [0,+\infty]$. This corresponds to a change of enriching category: from $([0,1],\leq,\otimes,1)$ to $([0,+\infty],\geq,+,0)$. Conceptually, we are transforming the category $\mathcal{C}_T$ into a generalized metric space, and the composition of morphisms witnesses the triangle inequality (see (4)).

This still leaves us with one problem: we only have morphisms between two different graded parts and we would like to have distance among any pair of objects in $\mathcal{C}_T$. This can be easely solved using the Yoneda embedding $\mathcal{C}_T^{op} \to \widehat{\mathcal{C}_T} = \mathcal{C}_T^{\mathbb{R}_+}$ sending each object $t \mapsto y^t$ to the functor it represents. As shown in (1), this functor is an isometric embedding which means that we have a way to compute any distance between objects in $\mathcal{C}_T$. In particular, given two words v and w in $\mathcal{C}_T^1$ we define:

$$d_{\mathcal{C}_T}(v,w) = d_{\widehat{\mathcal{C}_T}}(y^v, y^w) = \sup_{c \in \mathcal{C}_T} \{|y^v(c) - y^w(c)|\}. \tag{4}$$

The embedding allows to create the tensor of all the distances between any two objects. In particular, it allows building a matrix $D = (d_{ij})$ with the distances between words which is essentially what is needed to perform word embeddings. One interesting property of having distances defined in this way is that $d(x,z) = d(x,y) + d(y,z)$ since we are traversing a graph from the node x to the node z.

Lemma 3.4. *In $\mathcal{C}_T$, compostion of morphism yields $p(z|x) = p(z|y)p(y|x)$ or equivalently $d(x,z) = d(x,y) + d(y,z)$.*

Proof. We have that

$$p(z|y)p(y|x) = \frac{C(z)}{C(y)}\frac{C(y)}{C(x)} = \frac{C(z)}{C(x)} = p(z|x)$$

□

This lemma implies we have some sort of linearity of the distances or similarities. This is due to the fact that we can consider an enriched category as a weighted graph where the distance between two nodes is the sum (or composition) of distances. This adds a "rigidity" to the space C_T^1 making a configuration of points in a metric space where there is some linearity between the embedded points. It will be heavily used in the following section.

4 The Word2Vec Algorithm

In this section we study the CBOW and Skip-gram algorithms of (7). With that we analyze how they produce vector encodings of words with certain properties. One thing to note is that

there is a clash of notation between (1) and what is done in machine learning. In machine learning given an n-gram g one would compute $p(w|g) = p(w|g^-)$ with $g = g^- w$. This means the probability that w is the next word given a context g^-. However, in the category $\mathcal{C}_T$ we can only compute $p(y|x)$ if the $x \leq y$. For this reason, we fix the notation that $p(w|g) := p(g'|g)$ with $g' - g = w$.

4.1 The CBOW and Skip-gram algorithms

In (7), the authors introduced two architectures to compute word embeddings: CBOW and continuous Skip-gram (see Figure 1). Both algorithms are trained to predict words: CBOW is used to predict the middle word of a sequence and Skip-gram is used to predict the surrounding words (or two sequences in a text) of a given word.

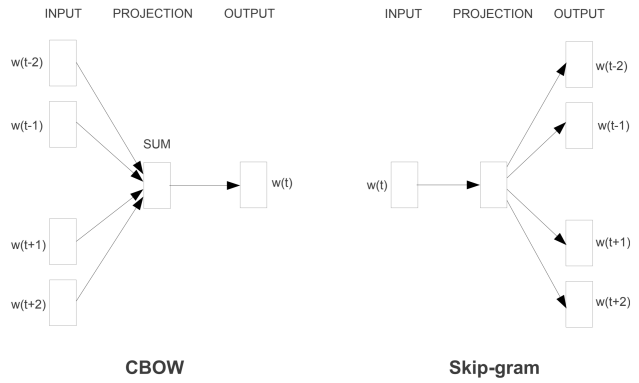

Figure 1: Architectures of the CBOW and Skip-gram algorithms.

Merely by the shape of the diagrams, we can conclude that the CBOW algorithm resembles a colimit diagram and the Skip-gram (being its dual) resembles a limit diagram. In particular, the predicted word or surrounding sequences should fit into pushout and pullback diagrams of $wg^+ \to g \leftarrow wg^-$ and $wg^+ \leftarrow w \to wg^-$ where $g^+ = w_{t+1:t+k}$, $g^- = w_{t-k:t-1}$ and $g = g^- w g^+$.

Here, we are considering the enriched version of the limit and colimit diagrams which, by their universal properties would imply that the chosen word or surroundings sequences maximize the conditional probability or, equivalently, minimize the distance. Of course, due to the structure of the category $\mathcal{C}_T$, these diagrams are not actually depicting what the algorithm is doing since we do not have morphism between objects in the same graded piece. However, this can be solved by passing to the enriched category of copresheaves $\widehat{\mathcal{C}_T} = \mathcal{C}_T^{\mathbb{R}_+}$ via the Yoneda embedding. In this setting, the above diagrams are replaced by the diagrams with the functors represented by the objects:

$$\begin{array}{ccc} y^w & \longrightarrow & y^{g^- w} \\ \downarrow & \ulcorner & \downarrow \\ y^{wg^+} & \longrightarrow & y^g \end{array} \qquad \begin{array}{ccc} y^w & \longrightarrow & y^{g^- w} \\ \downarrow & & \downarrow \\ y^{wg^+} & \longrightarrow & y^g \end{array}$$

Thus, the algorithms are trained to produce the pullback and pushout diagrams. Now that we understand categorically what the algorithms are doing only one thing remains to be explained: how these algorithms yield and embedding of $\mathcal{C}_T^1$ into a real vector space of dimension N that can be, and often is, different from $|\mathcal{C}_T^1|$. In order to answer that question we need to take into account the encoding used to work computationally.

Lastly, in order to be able to write the situation of the algorithm categorically we need to give the definition.

Definition 4.1. *Let T be a text and $\mathcal{C}_T$ its syntax category we let g^+ and g^- objects in $\mathcal{C}_T$. We define $p(w|g^{\pm}) := p(g^- w g^+|w)$, that is, the probability of the map $w \mapsto g^- w g^+$.*

4.2 The word embedding

Before studying the word embedding given by the Word2Vec algorithm we need to recall some basic definitions of Game Theory. For our purposes, 1-shot games with pure strategies will suffice.

Definition 4.2. *An n-**player game** is the data of a set X of $n \in \mathbb{N}$ elements with an associated finite set of pure strategies S_x for each player $x \in X$. To each player in X we associate a function:*

$$u_x \colon \prod_{x \in X} S_x \to \mathbb{R} \tag{5}$$

*called the **utility or payoff function**.*

Remark 4.3. *The key part of the definition is that the payoff function of each player depends on the strategies of all the players of the game.*

The next thing to note is that in an n-player game, given a tuple of strategies $s = (s_1, \ldots, s_n) \in \prod S_x$, the j-th player may want to change from strategy s_j to strategy t_j. From now on we will denote players by the letters i, j, k. We denote the new tuple of strategies by $(s; t_j)$. With this, we can give the definition of a Nash equilibrium point.

Definition 4.4. *Given an n-player game, an **equilibrium point** is an n-tuple $s = (s_1, \ldots, s_n) \in \prod S_x$ such that: $u_i(s) = \max\{u_i(s, t_i)\}$ for all $i \in X$.*

Thus an equilibrium point is an n-tuple such that each player maximizes its payoff if the strategies of the other player are fixed. This is a way of saying that no player has any incentive to change strategies. One important thing to note is that the equilibrium point may not be unique. For a more detailed exposition on the topic see (8).

We can apply this concept to the problem of embedding a weighted graph into $\mathbb{R}^N$ for some $N \in \mathbb{N}$.

Proposition 4.5. *Let X be a set of n points and $N \in \mathbb{N}$. Assume we have a matrix D of pairwise distances of elements of X. Then using metric multidimensional scaling (mMDS) to embed the set X with the least error between distances, in $\mathbb{R}^N$ is equivalent to computing the equilibrium point of an n-player game.*

Proof. We denote the elements of X by x_i with $i = 1, \ldots, n$. With this the distance between x_i and x_j is given by the entry d_{ij} of the matrix D. Metric multidimensional scaling is an algorithm that embeds $x_i \mapsto v_i$ minimizing:

$$\text{Stress}(v_1, \ldots, v_n) = \sum_{i=1}^{n} \sum_{i \neq j = 1, \ldots, n} (d_{ij} - \|v_i - v_j\|)^2. \tag{6}$$

With this, we can define an n-player game with payoff functions

$$u_i(s_1, \ldots, s_n) = - \sum_{i \neq j = 1, \ldots, n} (d_{ij} - \|s_i - s_j\|)^2 \tag{7}$$

Thus, s is an equilibrium point if and only if, each $u_i(s_1, \ldots, s_n)$ is maximal, which means that each summand of Stress is minimal yielding a solution of the metric multidimensional scaling.

On the other hand, if we have a tuple $v = (v_1, \ldots, v_n)$ such that $\text{Stress}(v_1, \ldots, v_n)$ is minimal. Then each $u_i(v_1, \ldots, v_n)$ is maximal because if it were not, there would exist a player k and a strategy w_k such that $u_k(v; w_k) \geq u_k(v)$. This implies that one of the summands of Stress could be smaller and in turn, Stress is not minimal contradicting the minimality of $\text{Stress}(v)$.

□

Remark 4.6. *The previous proposition implies that, given a text T and its syntax category $\mathcal{C}_T$, embedding the 1-graded piece into an N dimensional real vector space maintaining the distances (or the probabilities) is equivalent to finding equilibrium points of a V-player game where the payoff functions measure the difference of what can be seen in $\mathcal{C}_T$ and what is represented.*

To train the algorithms, the authors of (7) used 1-hot encodings for the space of words: each word in C_T^1 corresponds to a vector of the canonical basis of the $|\mathcal{C}_T^1|$-dimensional $\mathbb{R}$ vector space. Then, for the training part, they used a projection matrix W and a softmax function in the output layer. After training, the desired vector for the i-th word is the i-th row of the projection matrix W.

The simplest case is when the dimensions match, that is when the projection matrix is square. In this case, the embedding of the algorithm just corresponds to the Yoneda embedding.

Lemma 4.7. *Let T be a text and $\mathcal{C}_T$ its syntax category considered as a generalized metric space. Any perturbation of the distances between objects of $\mathcal{C}_T^1$ implies a perturbation of conditional probabilities.*

Proof. Let v, w objects in $\mathcal{C}_T^1$. Let $d = d(v, w)) \sup_{g \in \mathcal{C}_T} |d(v, g) - d(w, g)|$ the distance between these two words. If this distance is perturbed $\tilde{d} = d \pm \varepsilon$ then $\sup_{g \in \mathcal{C}_T} |d(v, g) - d(w, g)|$ changes. This means one of the distances (or both) gets perturbed. Let's assume only $d(v, g)$ changes. This means that $\tilde{d}(v, g) = d(v, g) \pm \varepsilon'$. Thus, applying the function $x \mapsto e^{-x}$ we get $\tilde{p}(w|g) = e^{-\tilde{d}(v,g)} = p(v|g) \cdot e^{\pm \varepsilon}$. □

Theorem 4.8. *Let T be a text and $\mathcal{C}_T$ the category of sequences. The CBOW algorithm witnesses the enriched version of the Yoneda embedding for $\mathcal{C}_T^1$ restricted to $\mathcal{C}_T^n$. This means, each object w in C_T^1 is sent to y^w which ca be regarded as a function $y^w \colon C_T^n \to \mathbb{R}_+$ via $y^w(g) = d(w, g)$.*

Proof. We will consider the case of CBOW since Skipgram is the dual algorithm. Let $V = |\mathcal{C}_T^1|$. For this proof, it is better to consider the category $\mathcal{C}_T$ enriched over $[0,1]$. That is to consider $\mathcal{C}_T$ as a probability space.

Considering 1-hot encodings of $\mathcal{C}_T^1$ we get that the context $w_{n-k:n-1} = g^-$ and $w_{n+1:n+k} = g^+$ are vectors with ones in the positions of the words w_i: $v^\pm$. Then the algorithm computes a vector v combining the parts $v^\pm$ (in the linear part). This is done in two parts, first for each context $v^\pm$ we compute a matrix/ vector multiplication with a first matrix W_1 and then we multiply the result by a second matrix W_2 to get the vector v. Finally, the algorithm has a softmax function $z = \sigma(v)$ and the target word is the one corresponding to the 1-hot encoding of the maximum entry of z. Thus, the algorithm can be summarized as follows (up to a constant):

$$\sigma(W_2 W_1(v^+ + v^-)). \tag{8}$$

The second matrix W_2 acts as the matrix of vectors of $\mathcal{C}_T^1$ and the first matrix W_1, which is the one that yields the embedding, acts as a "metric dualizer".

Notice that, a consequence of Proposition 4.5 having an embedding from $\mathcal{C}_T^1$ into $\mathbb{R}^N$ preserving (as best as possible) the distances given by

$$d_n(v,w) = d_n(y^v, y^w) = sup_{g \in \mathcal{C}_T^n} |y^v(g) - y^w(g)| \tag{9}$$

is equivalent to preserve the conditional probabilities $p(v|g)$ observed in $\mathcal{C}_T$ since we pass from one to the other via $-\log$.

Now, the softmax function $\sigma\colon \mathbb{R}^V \to [0,1]^V$:

$$\sigma(x) = \left(\frac{\exp(x_1)}{\sum_{i=1}^n \exp(x_i)}, \ldots, \frac{\exp(x_n)}{\sum_{i=1}^n \exp(x_i)} \right)$$

computes the relative probabilities given the intput $g^\pm$ of $p(e_i|g^\pm)$.

Finally, the loss function of the algorithm is the cross-entropy loss:

$$H(p,q) = \sum_{v \in \mathcal{C}_T^1} p(v|g^\pm) q(v|g^\pm)$$

where p is the probability distribution observed in the data set and q is the probability distribution given by the algorithm which is the result of $\sigma(W_2 W_1(v^+ + v^-))$. This function can be written as follows:

$$H(p,q) = H(p) + D_{KL}(p\|q) \tag{10}$$

where $H(p)$ is the entropy of the probability distribution p and $D_{KL}(p\|q)$ is the Kullback-Leiber dvergence:

$$D_{KL}(p\|q) = \sum_{v \in \mathcal{C}_T^1} p(v|g^\pm) \log \left(\frac{p(v|g^\pm)}{q(v|g^\pm)} \right). \tag{11}$$

The entropy $H(p)$ cannot be modified by the algorithm since it comes directly from the category $\mathcal{C}_T$. Furthermore, $D_{KL}(p\|q) \geq 0$ and $D_{KL}(p\|q) = 0$ if and only if $p = q$ as probability distributions. Thus, minimizing $H(p\|q)$ is equivalent to minimizing $D_{KL}(p\|q)$ which, by Lemma 4.7, is equivalent to minimizing the differences of distances. Hence, embedding $\mathcal{C}_T^1$ into $\mathbb{R}^N$ via multidimensional scaling is equivalent to embedding $\mathcal{C}_T^1$ into $\mathbb{R}^N$ via the CBOW algorithm. □

Remark 4.9. *1. In the last part of the proof, the two methods being equivalent means that the embedding achieved via the CBOW algorithm is a solution of the mMDS and that from the solution of the multidimensional scaling we can obtain a probability distribution that minimizes the Kullback-Leiber divergence and hence the cross-entropy.*

2. In (3), the author explains that the most powerful way to distinguish between the two distributions p and q based on an observation w (drawn from one of them) is through the log of the ratio of their likelihoods: $\log(p(w)) - log(q(w))$ *which is what we get if we compare the distances. This means that the most powerful way to distinguish the distribution is also a very intuitive one since we can picture them closer or further apart.*

3. The algorithm uses specifically maps of the form $w \mapsto g^- w g^+$ *to compute the distances which in turn correspond to the embedding:* $(\mathcal{C}_T^1)^{op} \to \widehat{(\mathcal{C}_T^1)}_{|2k,\pm}$.

Theorem 4.8 can be summarized as the commutativity of the diagrams:

$$\begin{array}{ccccc} \mathcal{C}_T^{2k,\pm} & & & & \\ {\scriptstyle mMDS}\downarrow & \searrow^{CBOW} & & & \\ R^N & \xrightarrow[like.]{} & R^V & \xrightarrow{\sigma} & [0,1]^V \cong \widehat{\mathcal{C}_T^1} \end{array} \qquad \begin{array}{ccccc} \mathcal{C}_T^1 & & & & \\ {\scriptstyle mMDS}\downarrow & \searrow^{Skip-gram} & & & \\ R^N & \xrightarrow[like.]{} & R^V & \xrightarrow{\sigma} & ([0,1]^V)^{2k} \cong \widehat{\mathcal{C}_T^{2k,\pm}} \end{array} \tag{12}$$

where $mMDS$ is the embedding given by the mMDS algorithm, *like.* is a function $\mathbb{R}^N \to \mathbb{R}^V$ giving some sort of likelihood between elements, σ is the softmax and $\mathcal{C}_T^{2k,\pm}$ being the set of pairs of sequences (g^-, g^+) of length k in T such that there exists w in $\mathcal{C}_T^1$ such that there is a map $w \mapsto g^- w g^+$. With this, we have another piece of the puzzle that we can append to the results of cited in Section 2 to achieve explainability of the machine learning algorithms via categorical methods.

5 Conclusions and further work

Analyzing the algorithms and datasets proposed in (7) has allowed us to understand how the proposed algorithms compute the vector representations of words. Furthermore, the added structure on $\mathcal{C}_T$ shed some light on the reasons for the linear/logical relations among vectors, e.g. $v_{king} - v_{man} + v_{woman} = v_{queen}$.

Proposition 4.5 yielded new properties of the embedding. Namely, the embedding obtained using mMDS or via the machine learning algorithms has the property of being stable: there is no incentive for any word to have any other vector representation provided the others are fixed. This ensures that the linear/logical relation among word vectors is preserved, as much

as possible, for every word. Thus, the embeddings are a faithful representation of the data: the text T.

Given the studied properties, one thing this can be applied to is bias detection. Indeed, if we restrict ourselves to the case of gender bias relating to one specific word, to check the bias one would simply compute the difference in the case of distances and the quotient in the case of probabilities of $y^{man}(w)$ and $y^{woman}(w)$. This is what is done via the word embeddings: see (6). However, the categorical perspective can be helpful in identifying gender biases not only between words but between the pair $(man, woman)$ and a context or series of contexts g of $\mathcal{C}_T$. By computing

$$\sup_{g \in G} |y^{man}(g) - y^{woman}(g)|$$

with G a subcategory of $\mathcal{C}_T$ we can estimate the gender bias present in those contexts. Furthermore, since we can restrict even further by considering only morphisms that send a particular word to a specific position of a sequence (by which we mean is the central word, final word, ...) we can refine even further the concept and get a finer metric. This is important since an asymmetry of the distances might not be a problem depending on the context: it should be reasonable that $y^{man}(pregnant) > y^{woman}(pregnant)$, that is the word "man" is is further from "pregnant" than the word "woman". With that in mind, one could propose changes to the data set or the embedding to reduce these biases. One possible way to achieve this is considering only certain morphisms in $\mathcal{C}_T$ to compute the distances or change the category. This is a line of future work.

Another possible research direction would be to study the functor:

$$\mathcal{C}_{(-)} \colon \mathcal{T} \to \mathcal{M} \tag{13}$$

that sends a text T to a generalized metric space $\mathcal{C}_T$. This could yield some interesting properties as well as a better understanding of how the enriched versions of the algorithms described in (2) are going to operate. Since NLP algorithms use some kind of word embedding, having a complete understanding of the pieces involved would help to have a safer implementation of those algorithms. This could be another line of future work.

Finally, the strategy outlined by equation (13) could be applied to other algorithmic tasks other than NLP. Indeed, the key ingredient of the framework is to have an adequate category of enrichment and a way (functor) that allows one to keep track of important information. Having this, one could explain categorically the behavior of learning algorithms thus helping us to gain knowledge of the problems we are solving with them.

6 Acknowledgments

This work is partially supported by the TAILOR project, a project funded by the EU Horizon 2020 research and innovation programme under GA No 952215, and by the GUARDIA project, a project funded by Generalitat Valenciana GVA-CEICE project PROMETEO/2018/002

References

[1] Tai-Danae Bradley, John Terilla & Yiannis Vlassopoulos (2022): *An enriched category theory of language: from syntax to semantics*. *La Matematica*, pp. 1–30.

[2] Brendan Fong, David Spivak & Rémy Tuyéras (2019): *Backprop as functor: A compositional perspective on supervised learning*. In: *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, IEEE, pp. 1–13.

[3] Solomon Kullback (1997): *Information theory and statistics*. Courier Corporation.

[4] F William Lawvere (1973): *Metric spaces, generalized logic, and closed categories*. *Rendiconti del seminario matématico e fisico di Milano* 43(1), pp. 135–166.

[5] Ulrike von Luxburg & Olivier Bousquet (2004): *Distance-Based Classification with Lipschitz Functions*. *J. Mach. Learn. Res.* 5(Jun), pp. 669–695.

[6] Sridhar Mahadevan (2022): *On The Universality of Diagrams for Causal Inference and The Causal Reproducing Property*. *arXiv preprint arXiv:2207.02917*.

[7] Tomas Mikolov, Kai Chen, G.s Corrado & Jeffrey Dean (2013): *Efficient Estimation of Word Representations in Vector Space*. *Proceedings of Workshop at ICLR* 2013.

[8] John Nash (1951): *Non-Cooperative Games*. *Annals of Mathematics* 54(2), pp. 286–295. Available at `http://www.jstor.org/stable/1969529`.

[9] Carlos Zednik (2021): *Solving the black box problem: a normative framework for explainable artificial intelligence*. *Philosophy & Technology* 34(2), pp. 265–288.