

# CS224W Homework 2

January 7, 2024

## 1 Node Embeddings with TransE [21 points]

While many real world systems are effectively modeled as graphs, graphs can be a cumbersome format for certain downstream applications, such as machine learning models. It is often useful to represent each node of a graph as a vector in a continuous low dimensional space. The goal is to preserve information about the structure of the graph in the vectors assigned to each node. For instance, the spectral embedding preserved structure in the sense that nodes connected by an edge were usually close together in the (one-dimensional) embedding  $x$ .

Multi-relational graphs are graphs with multiple types of edges. They are incredibly useful for representing structured information, as in knowledge graphs. There may be one node representing “Washington, DC” and another representing “United States”, and an edge between them with the type “Is capital of”. In order to create an embedding for this type of graph, we need to capture information about not just which edges exist, but what the types of those edges are. In this problem, we will explore a particular algorithm designed to learn node embeddings for multi-relational graphs.

The algorithm we will look at is TransE.<sup>1</sup> We will first introduce some notation used in the paper describing this algorithm. We’ll let a multi-relational graph  $G = (E, S, L)$  consist of the set of *entities*  $E$  (i.e., nodes), a set of edges  $S$ , and a set of possible relationships  $L$ . The set  $S$  consists of triples  $(h, \ell, t)$ , where  $h \in E$  is the *head* or source-node,  $\ell \in L$  is the relationship, and  $t \in E$  is the *tail* or destination-node. As a node embedding, TransE tries to learn embeddings of each entity  $e \in E$  into  $\mathbb{R}^k$  ( $k$ -dimensional vectors), which we will notate by  $\mathbf{e}$ . The main innovation of TransE is that each relationship  $\ell$  is also embedded as a vector  $\ell \in \mathbb{R}^k$ , such that the difference between the embeddings of entities linked via the relationship  $\ell$  is approximately  $\ell$ . That is, if  $(h, \ell, t) \in S$ , TransE tries to ensure that  $\mathbf{h} + \ell \approx \mathbf{t}$ . Simultaneously, TransE tries to make sure that  $\mathbf{h} + \ell \not\approx \mathbf{t}$  if the edge

---

<sup>1</sup>See the 2013 NeurIPS paper by Bordes et al: <https://papers.nips.cc/paper/5071-translating-embeddings-for-modeling-multi-relational-data.pdf>

$(h, \ell, t)$  does not exist.

**Note on notation:** we will use unbolded letters  $e, \ell$ , etc. to denote the entities and relationships in the graph, and bold letters  $\mathbf{e}, \ell$ , etc., to denote their corresponding embeddings. TransE accomplishes this by minimizing the following loss:

$$\mathcal{L} = \sum_{(h, \ell, t) \in S} \left( \sum_{(h', \ell, t') \in S'_{(h, \ell, t)}} [\gamma + d(\mathbf{h} + \ell, \mathbf{t}) - d(\mathbf{h}' + \ell, \mathbf{t}')]_+ \right) \quad (1)$$

Here  $(h', \ell, t')$  are "corrupted" triplets, chosen from the set  $S'_{(h, \ell, t)}$  of corruptions of  $(h, \ell, t)$ , which are all triples where either  $h$  or  $t$  (but not both) is replaced by a random entity, and  $\ell$  remains the same as the one in the original triplets.

$$S'_{(h, \ell, t)} = \{(h', \ell, t) \mid h' \in E\} \cup \{(h, \ell, t') \mid t' \in E\}$$

Additionally,  $\gamma > 0$  is a fixed scalar called the *margin*, the function  $d(\cdot, \cdot)$  is the Euclidean distance, and  $[\cdot]_+$  is the positive part function (defined as  $\max(0, \cdot)$ ). Finally, TransE restricts **all the entity embeddings to have length 1** :  $\|\mathbf{e}\|_2 = 1$  **for every**  $e \in E$ .

For reference, here is the TransE algorithm, as described in the original paper on page 3:

---

**Algorithm 1** Learning TransE

---

**input** Training set  $S = \{(h, \ell, t)\}$ , entities and rel. sets  $E$  and  $L$ , margin  $\gamma$ , embeddings dim.  $k$ .

```

1: initialize  $\ell \leftarrow \text{uniform}(-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}})$  for each  $\ell \in L$ 
2:    $\ell \leftarrow \ell / \|\ell\|$  for each  $\ell \in L$ 
3:    $\mathbf{e} \leftarrow \text{uniform}(-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}})$  for each entity  $e \in E$ 
4: loop
5:    $\mathbf{e} \leftarrow \mathbf{e} / \|\mathbf{e}\|$  for each entity  $e \in E$ 
6:    $S_{batch} \leftarrow \text{sample}(S, b)$  // sample a minibatch of size  $b$ 
7:    $T_{batch} \leftarrow \emptyset$  // initialize the set of pairs of triplets
8:   for  $(h, \ell, t) \in S_{batch}$  do
9:      $(h', \ell, t') \leftarrow \text{sample}(S'_{(h, \ell, t)})$  // sample a corrupted triplet
10:     $T_{batch} \leftarrow T_{batch} \cup \{(h, \ell, t), (h', \ell, t')\}$ 
11:   end for
12:   Update embeddings w.r.t.  $\sum_{((h, \ell, t), (h', \ell, t')) \in T_{batch}} \nabla [\gamma + d(\mathbf{h} + \ell, \mathbf{t}) - d(\mathbf{h}' + \ell, \mathbf{t}')]_+$ 
13: end loop
```

---

## 1.1 Simplified Objective [3 points]

Say we were intent on using a simpler loss function. Our objective function (1) includes a term maximizing the distance between  $\mathbf{h}' + \ell$  and  $\mathbf{t}'$ . If we instead

simplified the objective, and just tried to minimize

$$\mathcal{L}_{\text{simple}} = \sum_{(h,\ell,t) \in S} d(\mathbf{h} + \ell, \mathbf{t}), \quad (2)$$

we would obtain a useless embedding. Give an example of a simple graph and corresponding embeddings which will minimize the new objective function (2) all the way to zero, but still give a completely useless embedding.

**Hint:** Your graph should be non-trivial, i.e., it should include at least two nodes and at least one edge. Assume the embeddings are in 2 dimensions, i.e.,  $k = 2$ . What happens if  $\ell = \mathbf{0}$ ?

★ **Solution** ★ Minimizing the loss would effectively be reducing  $\ell$ , the distance between  $\mathbf{h}$  and  $\mathbf{t}$ , which would give the two entities  $\mathbf{h}$  and  $\mathbf{t}$  the same embedding. Example graph: entities  $\mathbf{h}$  and  $\mathbf{t}$  with connecting edge  $\ell$ .  $\mathbf{h}$  and  $\mathbf{t}$  both have embeddings of  $(0, 1)$  so  $\ell = \mathbf{0}$ , but the two entities are indifferentiable in the embedding space.

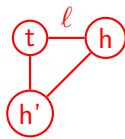
## 1.2 Utility of $\gamma$ [5 points]

We are interested in understanding what the margin term  $\gamma$  accomplishes. If we removed the margin term  $\gamma$  from our loss, and instead optimized

$$\mathcal{L}_{\text{no margin}} = \sum_{(h,\ell,t) \in S} \sum_{(h',\ell',t') \in S'_{(h,\ell,t)}} [d(\mathbf{h} + \ell, \mathbf{t}) - d(\mathbf{h}' + \ell', \mathbf{t}')]_+, \quad (3)$$

it turns out that we would again obtain a useless embedding. Give an example of a simple graph and corresponding embeddings which will minimize the new objective function (3) all the way to zero, but still give a completely useless embedding. By useless, we mean that in your example, you cannot tell just from the embeddings whether two nodes are linked by a particular relation (Note: your graph should be non-trivial, i.e., it should include at least two nodes and at least one edge. Assume the embeddings are in 2 dimensions, i.e.,  $k = 2$ .)

★ **Solution** ★ We would essentially be optimizing for  $d(\mathbf{h} + \ell, \mathbf{t}) = d(\mathbf{h}' + \ell', \mathbf{t}')$ . Let's say this distance value is  $D$ . Then Since, in each case, we are only perturbing one of  $\mathbf{h}$  or  $\mathbf{t}$ , WLOG perturb  $\mathbf{h}$  to become  $\mathbf{h}'$ . (See the graph below.) Then for some fixed  $\ell$ , the embeddings of  $\mathbf{h}$  and  $\mathbf{t}$  would be the same distance apart as those of  $\mathbf{h}'$  and  $\mathbf{t}$ . Since distance between two entity embeddings is what determines their relation type (or lack thereof), this loss function makes all distances between entity embeddings the same and makes relations indifferentiable. E.g.  $\mathbf{h} = (0,1)$ ,  $\mathbf{t} = (1, 0)$ ,  $\mathbf{h}' = (0, -1)$ . Then  $\ell = (\sqrt{2}, \sqrt{2})$  for both  $d(\mathbf{h}, \mathbf{t})$  and  $d(\mathbf{h}', \mathbf{t})$



### 1.3 Normalizing the embeddings [5 points]

Recall that TransE normalizes every entity embedding to have unit length (see line 5 of the algorithm). The quality of our embeddings would be much worse if we did not have this step. To understand why, imagine running the algorithm with line 5 omitted. What could the algorithm do to trivially minimize the loss in this case? What would the embeddings it generates look like?

★ Solution ★

### 1.4 Expressiveness of TransE embeddings [8 points]

Give an example of a simple graph for which no perfect embedding exists, i.e., no embedding perfectly satisfies  $\mathbf{u} + \boldsymbol{\ell} = \mathbf{v}$  for all  $(u, \ell, v) \in S$  and  $\mathbf{u} + \boldsymbol{\ell} \neq \mathbf{v}$  for  $(u, \ell, v) \notin S$ , for any choice of entity embeddings ( $\mathbf{e}$  for  $e \in E$ ) and relationship embeddings ( $\boldsymbol{\ell}$  for  $\ell \in L$ ). Explain why this graph has no perfect embedding in this system, and what that means about the expressiveness of TransE embeddings. As before, assume the embeddings are in 2 dimensions ( $k = 2$ ).

**Hint:** By expressiveness of TransE embeddings, we want you to talk about which type of relationships TransE can/cannot model with an example. (Note that the condition for this question is slightly different from that for Question 2.1 and what we ask you to answer is different as well).

★ Solution ★

## 2 Expressive Power of Knowledge Graph Embeddings [10 points]

TransE is a common method for learning representations of entities and relations in a knowledge graph. Given a triplet  $(h, \ell, t)$ , where entities embedded as  $h$  and  $t$  are related by a relation embedded as  $\ell$ , TransE trains entity and relation embeddings to make  $h + \ell$  close to  $t$ . There are some common patterns that relations form:

- Symmetry: A is married to B, and B is married to A.
- Inverse: A is teacher of B, and B is student of A. Note that teacher and student are 2 different relations and have their own embeddings.
- Composition: A is son of B; C is sister of B, then C is aunt of A. Again note that son, sister, and aunt are 3 different relations and have their own embeddings.

### 2.1 TransE Modeling [3 points]

For each of the above relational patterns, can TransE model it perfectly, such that  $h + \ell = t$  for all relations? Explain why or why not. Note that here  $\mathbf{0}$  embeddings for relation are undesirable since that means two entities related by that relation are identical and not distinguishable.

★ **Solution** ★ TransE cannot model relations of symmetry: For a symmetrical relation between  $h$  and  $t$ , if  $h + \ell = t$  where  $\ell \neq \mathbf{0}$ , then  $h = t + (-\ell)$ . By this same reasoning, we can see that TransE can model inverse relations. For relations of composition, where  $l_1 = l_2 l_3$  (e.g.  $l_{aunt} = l_{parent} l_{sister}$ ), TransE can model them: if  $a + \ell_{son} = b$  and  $c + \ell_{sis} = b$ , then  $c + (\ell_{sis} - \ell_{son}) = a$ .

### 2.2 RotatE Modeling [3 points]

Consider a new model, RotatE. Instead of training embeddings such that  $h + \ell \approx t$ , we train embeddings such that  $h \circ \ell \approx t$ . Here  $\circ$  means rotation. You can think of  $h$  as a vector of dimension  $2d$ , representing  $d$  2D points.  $\ell$  is a  $d$ -dimensional vector specifying rotation angles. When applying  $\circ$ , For all  $i \in 0 \dots d - 1$ ,  $(h_{2i}, h_{2i+1})$  is rotated clockwise by  $\ell_i$ . Similar to TransE, the entity embeddings are also normalized to L2 norm 1. Can RotatE model the above 3 relation patterns perfectly? Why or why not?

★ **Solution** ★ RotatE can model all three.

Symmetry: ( $\ell_{married} = 180^\circ$ ),

Inverse: ( $\ell_{teacher} + \ell_{student} = 360^\circ$  where  $\ell_{teacher} \neq \ell_{student}$ ),

Composition: ( $\ell_{son} + \ell_{sis} + \ell_{aunt} = 360^\circ$  where  $\ell_{son} \neq \ell_{sis} \neq \ell_{aunt}$ )

### 2.3 Failure Cases [4 points]

Give an example of a graph that RotatE cannot model perfectly. Can TransE model this graph perfectly? Assume that relation embeddings cannot be  $\mathbf{0}$  in either model.

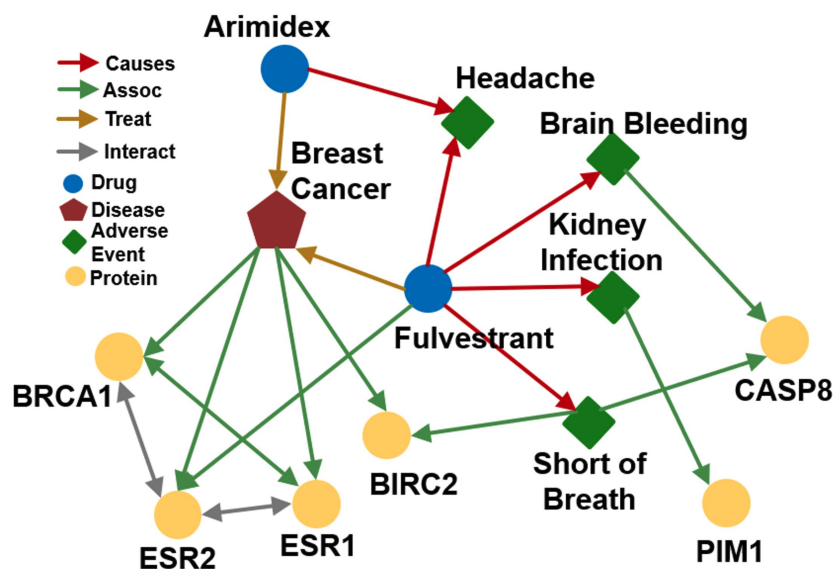
★ **Solution** ★ E.g. Transitive relation.  $S = \{a, b, c\}$  is a poset where  $a > b$  and  $b > c$ , thus we also have  $a > c$ . RotatE cannot both rotate  $\mathbf{a}$  by  $\ell_{>}$  degrees to get  $\mathbf{b}$  AND rotate  $\mathbf{a}$  by  $\ell_{>}$  degrees to get  $\mathbf{c}$  WITHOUT  $\mathbf{b}$  and  $\mathbf{c}$  being mapped to the same point and having the same  $\ell_{>}$  between them equal zero. By this same logic for vectors instead of rotation degrees, neither can TransE model transitivity.

### 3 Queries on Knowledge Graphs [14 points]

Knowledge graphs (KGs) can encode a wealth of information about the world. Beyond representing the information using knowledge graphs, we can often derive previously unknown insights about entities and relations in the graphs. In this question, we will explore different approaches for reasoning over knowledge graphs. Recall from that lecture that we are interested in predicting **tail** nodes given (**head**, **relation**). We will use the same formulation throughout this question.

#### 3.1 Path Queries on Complete KGs [3 points]

Consider the biomedicine knowledge graph from lecture. Assume the question of interest is: “What proteins are associated with diseases treated by Arimidex?” Write the question in query form (eg. (e:AnchorEntity, (r:Relation))) and find the answer(s) to the query. Partial credit will be rewarded to correct intermediate steps.



★ Solution ★

#### 3.2 Conjunctive Queries on Complete KGs [1 point]

Consider the same biomedicine knowledge graph from before. Write a conjunctive query to which BIRC2 is the only answer using drugs as anchor entities. If such a query doesn't exist, provide a one-sentence explanation.

★ Solution ★

### 3.3 Incomplete KGs [2 points]

A major issue with direct traversals on knowledge graphs is that they are usually incomplete in reality. One solution is to encode entities, relations, and queries in an embedding space that meaningfully organizes information. We would then be able to impute missing relation links by considering all nearby points of the query embedding as answers to the query. From lecture, we learned that TransE embeddings can be used for this. Can you come up with a way to adopt DistMult embeddings, which uses bilinear modeling, for answering path queries? If yes, describe in one or two sentences what can be modified from the TransE application. If no, provide a one-sentence explanation.

★ Solution ★

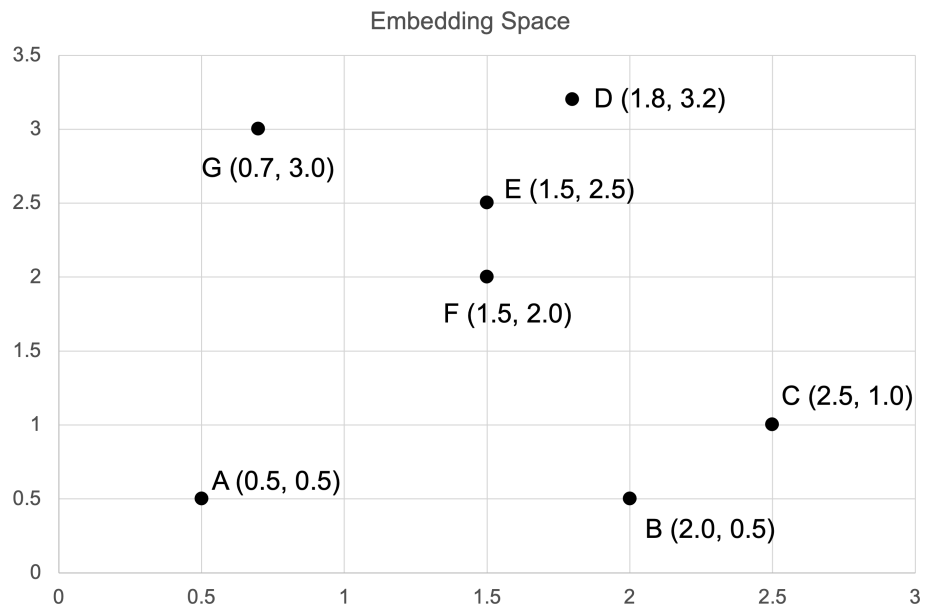
### 3.4 Query2box [8 points]

Query2box is an effective approach for answering complex conjunctive queries. Consider the following 2-dimensional embedding space. Assume that there are 7 entities  $A, B, C, D, E, F, G \in V$ , whose embeddings are shown below. There are 3 relations:  $R_1, R_2, R_3$ .  $R_1 \in R$  shifts the center of a box by  $(0.25, 2)$  and increases the width and height of a box by  $(0.5, 2)$ .  $R_2$  shifts the center of a box by  $(1, 0)$  and increases the width and height of a box by  $(1, 0)$ .  $R_3$  shifts the center of a box by  $(-0.75, 1)$  and increases the width and height of a box by  $(1.5, 3)$ .

Use the Query2box projection operator to find the answers to the conjunctive query:  $((e:A, (r:R_1, r:R_2), (e:C, (r:R_3)))$ . Show your work. Partial credit will be rewarded to correct intermediate steps.

Note: Shifting by a negative value means moving towards the left or bottom. Increasing the width and height by an amount means adding that amount in absolute value, not multiplying that amount as a factor. Assume that each path query starts with a box centered at the anchor entity with zero width and height.





★ Solution ★

## 4 Subgraph and Order Embeddings [20 points]

In lecture, we demonstrated that subgraph matching can be effectively learned by embedding subgraphs into the order embedding space. The reason is that many properties associated with subgraphs are naturally reflected in the order embedding space.

For this question, we say “graph  $A$  is a subgraph of graph  $B$ ” when there exists a subgraph of  $B$  that is graph-isomorphic to graph  $A$ . We additionally only consider the induced subgraph setting introduced in lecture, and all the order embeddings are non-negative.

Recall that the order embedding constraint states that:  $A$  is a subgraph of  $B$  if and only if  $z_A[i] \leq z_B[i]$  for all embedding dimension  $i$ . For simplicity, we do not consider anchor nodes in this question, and assume that the order embedding  $z_A$  is an embedding of graph  $A$ .

### 4.1 Transitivity [4 points]

Show that the subgraph relation is transitive: if graph  $A$  is a subgraph of graph  $B$ , and graph  $B$  is a subgraph of  $C$ , then graph  $A$  is a subgraph of  $C$ . The proof should make use of the subgraph isomorphism definition: if graph  $A$  is a subgraph of graph  $B$ , then there exists a bijective mapping  $f$  that maps all nodes in  $V_A$  to a subset of nodes in  $V_B$ , such that the subgraph of  $B$  induced by  $\{f(v)|v \in V_A\}$  is graph-isomorphic to  $A$ . (NOTE: You can assume that the composition of two bijective functions is bijective)

★ Solution ★

### 4.2 Anti-symmetry [4 points]

Use the same definition on subgraph isomorphism to show that the subgraph relation is anti-symmetric: if graph  $A$  is a subgraph of graph  $B$ , and graph  $B$  is a subgraph of graph  $A$ , then  $A$  and  $B$  are graph-isomorphic.

Hint: What do these conditions imply about the number of nodes in  $A$  and  $B$ ? How does this relate to graph isomorphism?

★ Solution ★

### 4.3 Common Subgraphs [3 points]

Consider a 2-dimensional order embedding space. Graph  $A$  is embedded into  $z_A$ , and graph  $B$  is embedded into  $z_B$ . Suppose that the order embedding constraint is perfectly preserved in this order embedding space (this is equivalent to assuming that  $z_A \preceq z_B$  holds if and only if  $A$  is a subgraph of  $B$ ). Prove that graph  $X$  is a

common subgraph of  $A$  and  $B$  if and only if  $z_X \preceq \min\{z_A, z_B\}$ . Here  $\min$  denotes the element-wise minimum between two embedding vectors.

★ Solution ★

#### 4.4 Order Embedding Constraints [3 points]

Suppose that graphs  $A, B, C$  are non-isomorphic graphs that are not subgraphs of each other. We embed them into a 2-dimensional order embedding space. Without loss of generality, suppose that we compare the values of their embeddings in the first dimension (dimension 0) and have  $z_A[0] > z_B[0] > z_C[0]$ . What does this imply about the relation among  $z_A[1], z_B[1], z_C[1]$ , assuming that the order embedding constraint is perfectly satisfied?

★ Solution ★

#### 4.5 Subgraph Relations [6 points]

In this question, we show that a 2-dimensional order embedding space is not sufficient to perfectly model subgraph relations.

Consider three non-isomorphic graphs:  $A$ ,  $B$ , and  $C$ . These graphs are not subgraphs of each other. Let's assume, without loss of generality, that  $z_A[0] > z_B[0] > z_C[0]$ . Now, imagine we have three other graphs:  $X$ ,  $Y$ , and  $Z$ . Each of these is a common subgraph of one or more of the original graphs ( $A$ ,  $B$ , or  $C$ ). For instance,  $X$  could be a common subgraph of both  $A$  and  $B$  and not  $C$ . The task is to construct a scenario where the embeddings of these subgraphs ( $X$ ,  $Y$ , and  $Z$ ) implicitly satisfy the conditions:  $z_X \preceq z_Y$  and  $z_X \preceq z_Z$ . You don't need to provide the specific embedding coordinates. Just describe the relationships between the subgraphs ( $X$ ,  $Y$ , and  $Z$ ) and the original graphs ( $A$ ,  $B$ , and  $C$ ). Also, explain why your example meets the given conditions (i.e.  $z_X \preceq z_Y$  and  $z_X \preceq z_Z$ ).

*Note that this condition implies that  $X$  is a common subgraph of  $Y$  and  $Z$ . However, one can construct actual example graphs of  $A, B, C, X, Y, Z$  such that  $X$  is not a common subgraph of  $Y$  and  $Z$ . This means that 2-dimensional order embedding space cannot perfectly model subgraph relations. Hence in practice, we use high-dimensional order embedding space. For this question, you do not have to show such example graphs.*

★ Solution ★

## 5 Honor Code [0 points]

(X) I have read and understood Stanford Honor Code before I submitted my work.

**\*\*Collaboration:** Write down the names & SUNetIDs of students you collaborated with on Homework 2 (None if you didn't).**\*\***

**\*\*Note:** Read our website on our policy about collaboration!**\*\***