Chapter 5

Description of the implementation and the experiments

This chapter will provide a comprehensive description of the challenges encountered. We will provide a comprehensive analysis of each pipeline phase. All of the preprocessing steps, models, and evaluation code are on GitHub[1].

1.1 Pipeline

As explained in 4.1.3, the main goal of this thesis is to use triple classification to predict relationships and automatically test the newly suggested triples. Figure 5.1 illustrates the pipeline that addresses the problem. Initially, the KG was stored in turtle format. Following this is the preparatory phase, which consists of two tasks: generating negative triples and preparing the data for the model. We then send the triples to the machine-learning model. For the models, TensorFlow was utilized. The evaluation phase follows, where we assess the model's performance on the test set and generate graphical representations. Subsequently, we initiate an additional, one-time procedure. During this stage, we deliberately choose pairings of nodes from the graph that do not share a direct connection through a relationship. For each experiment, we employ these pairings in order to forecast the relationship using each model. We attempt to perform an automated evaluation of the new triples in the final phase.

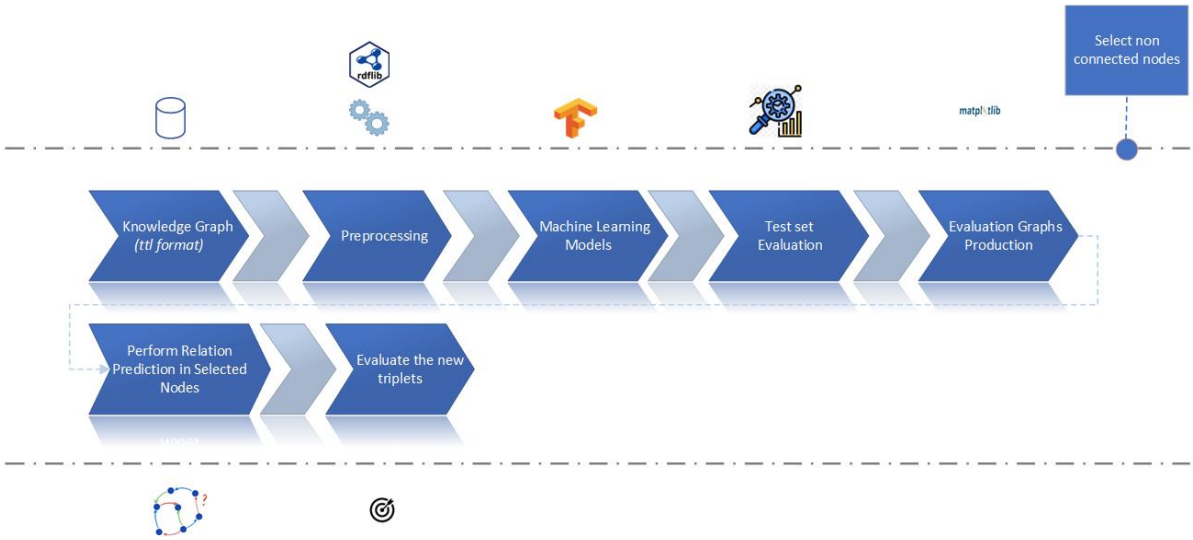


Figure 5.1: Pipeline followed

[1] https://anonymous.4open.science/r/Repo-67B2/README.md

1.2 Preprocessing

The function *preprocess_data* performs the preprocessing, transforming a triple-representation graph into a structured format for computational analysis, as shown in Figure 5.2. This transformation includes the extraction and indexing of unique entities and relations, the creation of mapping dictionaries, and the conversion of the original triples into indexed form.

Initially, the method detects and isolates all distinct entities present in the graph. The method retrieves entities from the triples' subject and object locations. We accomplish this by first establishing a collection of themes and a collection of objects, then merging these collections together. Simultaneously, the function retrieves all distinct relationships from the graph, identified based on the predicate position of the triples.

Subsequently, the function creates many mapping dictionaries to simplify the conversion process between entities and indices. More precisely, by iterating over the list of unique entities, it generates a dictionary named $entity2idx$ that assigns a distinct integer index to each distinct entity. Furthermore, it generates a reverse mapping ($idx2entity$) to link each integer index to its corresponding entity, a valuable tool for decoding or comprehending the results of computational procedures. We build a dictionary named $relation2idx$ to establish a one-to-one correspondence between each unique relation and a unique integer index. Furthermore, we establish a reverse mapping known as $idx2relation$ to link each integer index to its corresponding relation.

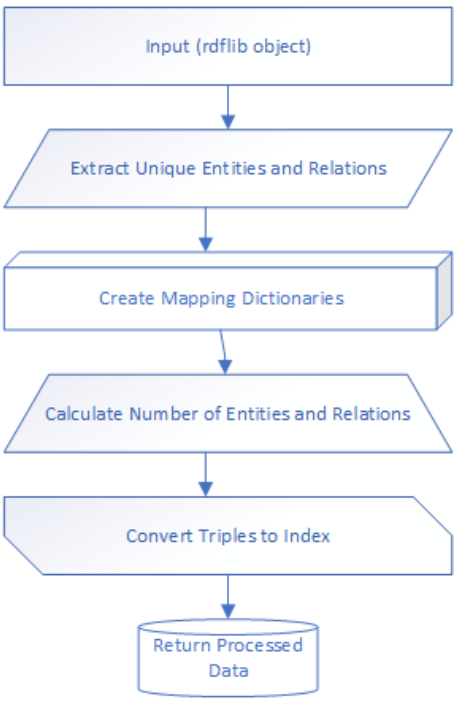


Figure 5.2: Preprocessing function

The function creates the mappings and then transforms the original graph triples into a different format by substituting each item and relation with its associated index. The outcome is a collection of triplets, with each member being a tuple of numbers that represents the

indices of the subject, predicate, and object. The indexed format is well-suited for a range of computational approaches, including machine learning algorithms.

In the end, the function provides the following information: the number of unique entities ($num\_entities$) and relations ($num\_relations$), the set of indexed triples ($triples$), and the mappings between entities and their corresponding indices ($entity2idx$ and $idx2entity$) and relations and their corresponding indices ($relation2idx$ and $idx2relation$). These outputs make sure that the graph data is in the consistent number format needed for algorithmic processing. This enables the effective and efficient use of graph data in computational models.

Technically, the data is now ready to feed into the model. However, the triples now only contain positive values. If we train the model this way, it will predict a very high probability for every triple it sees. To address this, we created some negative triples. We use the following methods to create these negative triples:

Method 1: Altering the object

Most experiments have applied this simple yet effective method. It involves the following steps:

1. **Iterate Over Original Triples:** The function goes over every triple (s, p, o) in the collection of initial triples. Every triple consists of an object o, a subject s, and a predication p.
2. **Random Object Replacement:** For every triple, we disturb the object o to produce a corrupted triple. This is accomplished by adding to the original object index o a random number between 1 and $numentities - 1$. The addition guarantees corruption because it ensures that the new index does not match the old object.
3. **Modulo Operation:** The addition then produces a modulo $numentities$ value that guarantees the new object index falls inside the appropriate range of entity indices. This stage is critical in preventing index overflow and ensuring consistency in the entity index space.
4. **Formation of Negative Triples:** After that, we compile the corrupted triples (s, p, o') into a list where o' stands for the new object index at random. Assuming their absence in the original knowledge graph, we regard these triples as "negative" because of their artificial production.

The approach's advantages

1. **Simple Implementation:** The technique for producing negative triples is computationally efficient and easy to use. It uses computationally cheap operations—that is, fundamental arithmetic and random number generation.
2. **Control Over Corruption:** The method preserves some degree of control over the produced negatives by just altering the object and leaving the subject and predication unaltered. This concentrated corruption teaches the model that while subjects and predicates are still legitimate, the object may not always be accurate.

Disadvantages of the Approach

1. **Randomness Limitation:** Sometimes the randomization in object replacement generates negative triples that unintentionally exist in the original graph, thereby not significantly advancing the learning process. Throughout training, this unintentional creation of false negatives may be deceptive.
2. **Uniform Corruption:** The technique employs a uniform random corruption without regard for the graph's structural or semantic properties. This homogeneity could not always represent complex, incorrect connections, so less effective training might result.
3. **Potential Redundancy:** While the modulo operation guarantees that indices fall within range, improper handling of it may cause negative generation to show patterns. This could reduce the number of negative samples, thereby affecting the training's efficacy.

Method 2: Diverse negative sampling

To enhance the training of machine learning models for knowledge graph embeddings, we define the $diverse\_negative\_sampling$ function for this method. This function adds variety to the negative samples by corrupting only certain parts (subject, predicate, and object) of the original triples based on the corruption strength that was given. It consists of the following steps:

1. **Initialization**:
   - We initialize an empty list of $negative\_triples$ to store the generated negative triples.
   - To ensure the uniqueness of negative triples, we convert the original triples into a set named $triple\_set$ for efficient lookup.
2. **Batch Processing**:
   - We process the triples in batches of $batch\_size$ to efficiently manage large datasets. For each batch:
     - We create a sublist $batch\_triples$ that contains the triples for the current batch.
     - We initialize an empty list, $batch\_negatives$, to store the negative triples generated for the current batch.
3. **Corruption Based on Strength**:
   - For each triple (s, p, o) in the batch, different corruption strategies are applied based on the specified $corruption\_strength$:
     - Low Corruption:
       - Only the subject s is replaced with a randomly selected entity index s_{new}.
       - A while-loop ensures that the newly generated triple (s_{new}, p, o) does not exist in the original triples.
     - Medium Corruption:
       - Both the subject s and the object o are replaced with randomly selected entity indices s_{new} and o_{new}, respectively.

        - Two negative triples (s_{new}, p, o) and (s, p, o_{new}) are generated.
        - While-loops ensure that neither of these new triples exists in the original triples.
    - High Corruption:
        - The subject s, predicate p, and object o are all replaced with randomly selected indices s_{new}, p_{new}, and o_{new}, respectively.
        - Three negative triples (s_{new}, p, o), (s, p_{new}, o), and (s, p, o_{new}) are generated.
        - While-loops ensure that none of these new triples exist in the original triples.
4. **Collecting Negative Triples**:
    - We append the generated negative triples for the current batch to the $negative\_triples$ list.
    - We repeat this process for all batches until we have processed all original triples.

5. **Return Negative Triples**:
    - The function returns the complete list of generated negative triples.

The approach's advantages

1. **Diversity in Negative Samples:** By changing the corruption strength, the function generates a variety of negative triples, allowing the model to learn more precisely to distinguish between valid and invalid triples.
2. **Controlled Corruption:** The ability to indicate the degree of corruption lets one optimize the training process. Early training phases could benefit from less corruption; yet, as the model becomes stronger, we can include more corruption.
3. **Efficiency in Handling Large Datasets:** Processing triples in batches makes the function appropriate for big-scale datasets as it helps control memory use and computing demand.
4. **Avoidance of Redundant Negatives:** Using while-loops to search for freshly produced triples guarantees that the negative samples are indeed unique and do not unintentionally match positive triples.

Disadvantages of the Approach

1. **Computational Overhead:** Particularly for big graphs with numerous entities and interactions, the while-loops employed to search for triples might impose processing cost. This might halt the negative sampling process.
2. **Potential for Infinite Loops:** Under high entity density and poor graph variety, the while-loops may take a considerable amount of time to identify a valid negative triple, or in extreme circumstances, they may cause endless loops.
3. **Uniform Randomness:** The random selection of new indices disregards the structural or semantic aspects of the network, making negative triples less useful for training.
4. **Batch Size Dependency:** The function's performance and efficiency may change depending on the batch size option. A wrong batch size could cause extended processing delays or ineffective memory use.

Method 3: Popular based sampling

For this method, we define the $popularity\_based\_sampling$ function, which generates negative triples by leveraging the frequency of entities and relations in the original dataset. This method mimics the graph's popularity distribution by giving more frequent entities and relations a higher chance of selection. It involves the following steps:

1. **Counting Entity and Relation Frequencies**:
   - The function initializes two counters: $entity\_counter$ and $relation\_counter$.
   - It iterates over the original triples to count the occurrences of each entity in the subject position and each relation.
2. **Ranking Entities and Relations**:
   - We rank entities and relations based on their frequency. The function uses $rankdata$ to assign ranks, where the most frequent entities and relations get the lowest ranks (indicating higher popularity).

3. **Probability Distribution Calculation**:
   - We calculate a probability distribution based on the ranks. The probability of selecting an entity or relation is inversely proportional to its rank plus one. This means that higher ranked (and more popular) entities and relationships have higher probabilities.
   - We normalize the raw probabilities to ensure their sum equals one. This normalization ensures that the resulting values form a valid probability distribution.
4. **Sampling Based on Probability Distribution**:
   - Entities and relations are sampled according to the calculated probability distributions. The function uses $np.random.choice$ to randomly select entities and relations based on their respective probabilities.
   - The sample size is determined by $batch\_size$, ensuring that a fixed number of negative triples are generated in each batch.
5. **Formation of Negative Triples**:
   - We pair the sampled entities and relations to form negative triples. We construct each triple by combining a sampled entity, a sampled relation, and another instance of the same entity as the object.
   - We return the resulting list of negative triples.

The approach's advantages

1. **Realistic Negative Samples**: Realism selects entities and links based on their popularity, thereby improving the production of negative triples. Actual knowledge graphs typically display asymmetrical popularity distributions, aligning with this distribution.
2. **Concentrate on Frequent Entities and Relations**: Negative triples are more likely to include more commonly occurring entities and linkages. Using this approach ensures that the model gains the capacity to control regular events, which are usually more relevant in terms of performance in practical situations.

3. **Probability-based Sampling:** Using a probability distribution ensures that negative samples are generated in a regulated, statistically sound manner. This habit reduces the likelihood of creating unimportant or unenlightening negative triples.
4. **Efficiency**: Processing large amounts of information in chunks allows the function to properly control memory use and computational load.

Disadvantages of the Approach

1. **Favoring Prominent Entities and Relationships**: While focusing on well-known things and interactions has benefits, this strategy may unintentionally lead to the neglect of less frequently occurring but equally important entities and relationships.
2. **Unbiased Object Selection**: For both the subject and object locations in negative triples, the current approach uses the same entity samples. This strategy may not introduce sufficient variety. This could restrict the positive value of the negative samples.
3. **Restricted Semantic Context**: Advocates of frequency alone ignore the semantic background of objects and relationships. A lack of context may result in negative triples that are less instructive and fail to sufficiently push the model through the training process.

Method 4: Bernoulli sampling

This method implements Bernoulli-negative sampling. First, we initialize an empty list named $negative\_triples$ to store the generated negative triples. Next, we iterate through each triple in the list of positive triples, $triples$, which consists of subject (s), predicate (p), and object (o). For each triple, we perform a Bernoulli trial by generating a random integer (0 or 1) using $np.random.randint(2)$. This simulates a coin toss, deciding whether to corrupt the subject or the object of the triple.

- $Random.choice(num\_entities)$ randomly selects a negative subject (s_neg) from the range of available entities if the coin toss result is 0. We then form a new triple by replacing the original subject s with this negative subject s_neg, while maintaining the predicate p and the object o unchanged. We append this new triple to the $negative\_triples$ list.
- Using $np.random.choice(num\_entities)$, we randomly select a negative object (o_neg) from the range of available entities if the coin toss result is 1. We form a new triple by replacing the original object with this negative object, o_neg, while maintaining the subject and predicate unchanged. We append this new triple to the $negative\_triples$ list.

By the end of the iteration, the $negative\_triples$ list contains a set of negative samples, each of which is a corrupted version of an original positive triple. This negative sampling approach randomly corrupts both subjects and objects, resulting in a balanced set of negative examples.

1.3 Evaluation

There are two phases to the model evaluation process. Like most machine learning models, there is a direct phase on the test set, followed by a relation prediction task that occurs after the model has undergone training and evaluation on the test set.

Test set Evaluation

We make the evaluation using five metrics suitable for binary classification problems. We use *precision*, *recall*, *F1 score*, *roc_auc* score, *pr_auc* score, and *Matthews Correlation Coefficient* (MCC). We use the random predicting method as the baseline method, which provides a basis for comparing the model results. In this method, we randomly choose negative or positive instances, so we get all metric values of 0.5. Next, we obtain various graphs that offer insights into the model's performance. For every model, we take the learning curve plot, the precision recall curve, the confusion matrix (with a threshold > 0.5), the calibration plot for the probabilities, and some plots that interpret the model. For the interpretation, we use the LIME (Local Interpretable Model-Agnostic Explanations) framework for the predictions.

Specifically, it explains how individual features (subject, predicate, and object) contribute to the model's predictions for a subset of instances from the test dataset. For these instances, we also visualize the mean feature importance. We select instances from the test set for analysis. The exact number of instances we explain—usually 300 to 1200—varies according to how much time the model needs to predict each instance. For some models, we also show the exact impact of the contribution of subject, predicate, and object for five randomly selected triples, as shown in Table 5.1. The evaluation mentioned above pertains to the test set and focuses on the triple classification task.

Table 5.1: The five randomly selected triples

$Triple\ 1$	$glossaryArticle118$	$ns1{:}\ hasReference$	$referenceSource59$
$Triple\ 2$	$hlth\_ehis\_aw1u$	$ns1{:}\ term$	"$hlth\_ehis\_aw1u$"
$Triple\ 3$	$paragraph9574\_3455$	a	$ns1{:}\ Paragraph$
$Triple\ 4$	$ns1{:}\ ei\_qna$	a	$ns1{:}\ StatisticalData$
$Triple\ 5$	$ns1{:}\ fats\_08$	$ns1{:}\ level$	"4"

Relation Prediction and Evaluation

After this evaluation, we perform relation prediction using the trained model for the triple classification task. To perform relation prediction, we require a pair of entities for which we will predict a single relation from the total number of relations present in the graph. We generate these pairs using the following method, as shown in Figure 5.3: We use random walks to sample entities from the graph and generate pairs of non-connected nodes, saving the results to a file. The $random\_walk$ function is defined to perform a random walk on the graph, starting with a given entity. It accepts three arguments: $graph, entity$, and $walk\_length$. The function initializes a walk, starting from the provided entity, and appends it to a list named walk. It then iterates up to the specified $walk\_length$, selecting a random neighbor of the current entity at each step. If the current entity has no neighbors, indicating

a dead end, the walk terminates early. We then return the walk and the resulting list, which includes the order of entities visited during the random walk.

Next, the $sample\_entities$ function is defined to sample a specified number of unique entities ($num\_samples$) from the graph using random walks. This function initializes an empty set named *entities* to store the unique entities encountered during the walks. Until it collects the desired number of unique entities, the function repeatedly selects a triple from the graph and randomly selects either the subject or object of the triple as the starting entity. The function then performs a random walk from the selected entity, updating the set of entities with those encountered during the walk. We convert the set to a list once it contains enough entities and return the first $num\_samples$ entities.

Then we proceed to generate pairs of non-connected nodes. The variable num_pairs determine the quantity of non-connected pairs for generation. To ensure enough diversity in the sampled entities, the $sample\_size$ is set to twice the number of pairs. We sample entities using the $sample\_entities$ function with a walk length of five. We initialize an empty list named $non\_connected\_pairs$ to store the generated pairs. We enter a loop to create pairs of non-connected nodes by randomly selecting two distinct entities from the sampled entities. If the two entities differ, they form a pair and add it to the $non\_connected\_pairs$ list. We continue this process until we obtain the required number of pairs (num_pairs). Finally, the pairs of non-connected nodes are saved to a file named non_connected_pairs.txt. The file is opened in write mode, and each pair is written in the format $entity1$ --- $entity2$.

In this case we choose to create five files with non-connected pairs containing 200, 200, 300, 300 and 500 pairs of nodes respectively. Now to perform the relation prediction we make the following. Initially, the $RDFLib$ library is used to load the RDF graph from a Turtle file. The graph is parsed, and a dictionary is created to store domain and range information for each relation by iterating through the triples in the graph. If a triple specifies a domain or range, the corresponding relation and entity are recorded in the $relation\_domain$ and $relation\_range$ dictionaries.

Afterwards, we define several functions to handle the relation prediction process, which is shown in Figure 5.4. The $generate\_candidate\_triples$ function creates candidate triples by pairing each node pair with all possible relations, producing a list of candidate triples. The $predict\_probabilities$ function then uses a provided model to predict the probabilities of these candidate triples, converting the list of triples to a NumPy array and utilizing the model's predict method to obtain probabilities.

The $select\_relations$ function selects the relation with the highest predicted probability for each node pair by iterating through the candidate triples' probabilities. It keeps track of the highest probability relation for each pair and stores the corresponding relation index. The $filter\_candidate\_triples$ function filters these candidate triples based on domain and range restrictions using the previously constructed dictionaries. It ensures that only valid triples, according to these restrictions, are retained.

After defining these functions, we process multiple files containing pairs of non-connected nodes. For each file, it reads the node pairs and maps non-integer values to unique integer identifiers to facilitate processing. It converts the node pairs to integers and generates

candidate triples. We filter these triples based on domain and range restrictions, then use the model to predict their probabilities. We select the highest probability relations for each node pair and write the results to an output file. We write the triples in a format that includes the probability, sorted in descending order of probability.

As the final step in this section, we aim to create an automated evaluation of the new triples we recommend. To achieve this, do the following, as shown in Figure 5.5: We load triples from files line by line, extracting the subject, predicate, object, and probability from each line using regular expressions. These extracted triples are stored in a list. Next, we extract unique entities and relations from the loaded triples, storing them in sets named entities and relations, respectively. Following this, the parameters for the $FastText$ model are defined, including the vector size, window size, minimum count, and the number of worker threads. The $FastText$ model is then trained on the triples to learn vector representations (embeddings) for the entities and relations.

A function named $get\_embedding$ is defined to retrieve the embedding for a given word (entity or relation) from the $FastText$ model. A zero vector is returned if the term is not detected in the model's vocabulary. This function is employed to acquire and store embeddings for all entities and relations in the dictionaries $entity\_embeddings$ and $relation\_embeddings$. These dictionaries are subsequently amalgamated to form a singular dictionary, $all\_embeddings$. Principal Component Analysis (PCA) helps visualize embeddings by reducing their dimensionality to two dimensions. This decrease helps the embeddings to be seen in a 2D graph. The embeddings that have been transformed by PCA are stored in the $pca\_embeddings$ variable.

We next compute the silhouette score for many values of k ranging from 2 to 10 to ascertain the ideal number of clusters (k) for K-means clustering. Higher scores indicate better-defined clusters; the silhouette score gauges how similar an item is to its own cluster relative to other clusters. The value of k that yields the highest silhouette score is selected as the optimal number of clusters. Using the best value of k, K-means clustering is performed on the original embeddings, and the resulting cluster labels are obtained.

Finally, we compute and prints three clustering quality metrics: the $silhouette\ score$, the $Davies - Bouldin\ index$, and the $Calinski - Harabasz\ index$. The $silhouette\ score$, as previously mentioned, indicates how well the clusters are defined. The $Davies - Bouldin$ index measures the average similarity ratio of each cluster with the cluster that is most like it, with lower values indicating better clustering. The $Calinski - Harabasz\ index$ evaluates the ratio of the sum of between-cluster dispersion to within-cluster dispersion, with higher values indicating better-defined clusters.

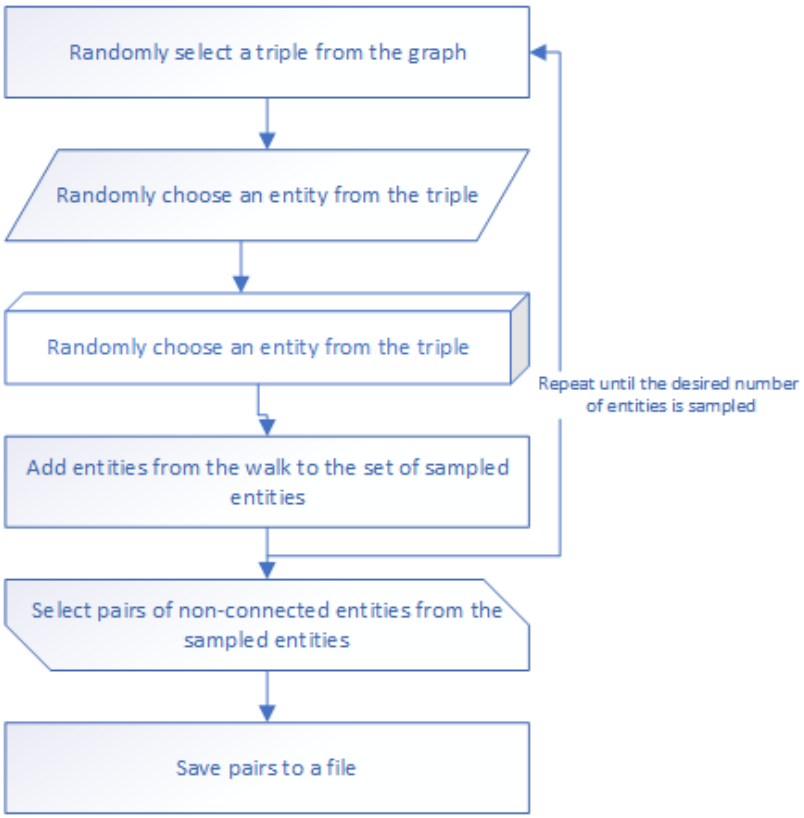


Figure 5.3: Production of the node pairs

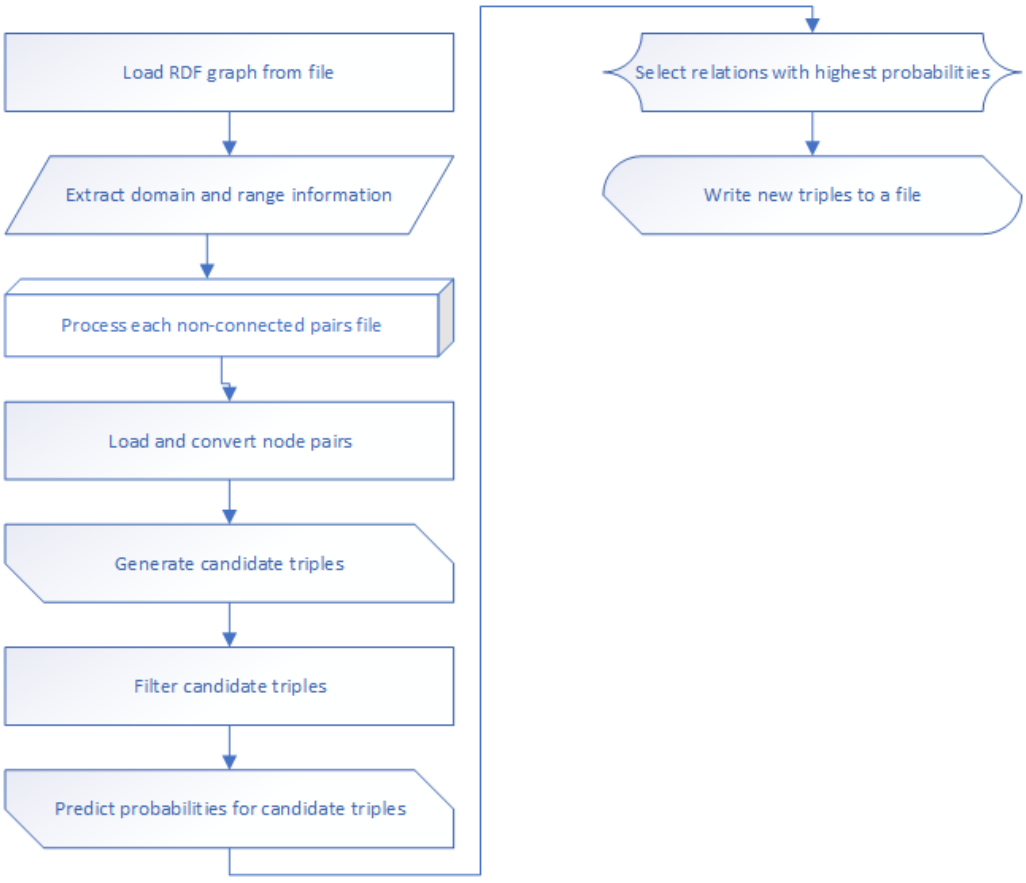


Figure 5.4:Relation prediction process

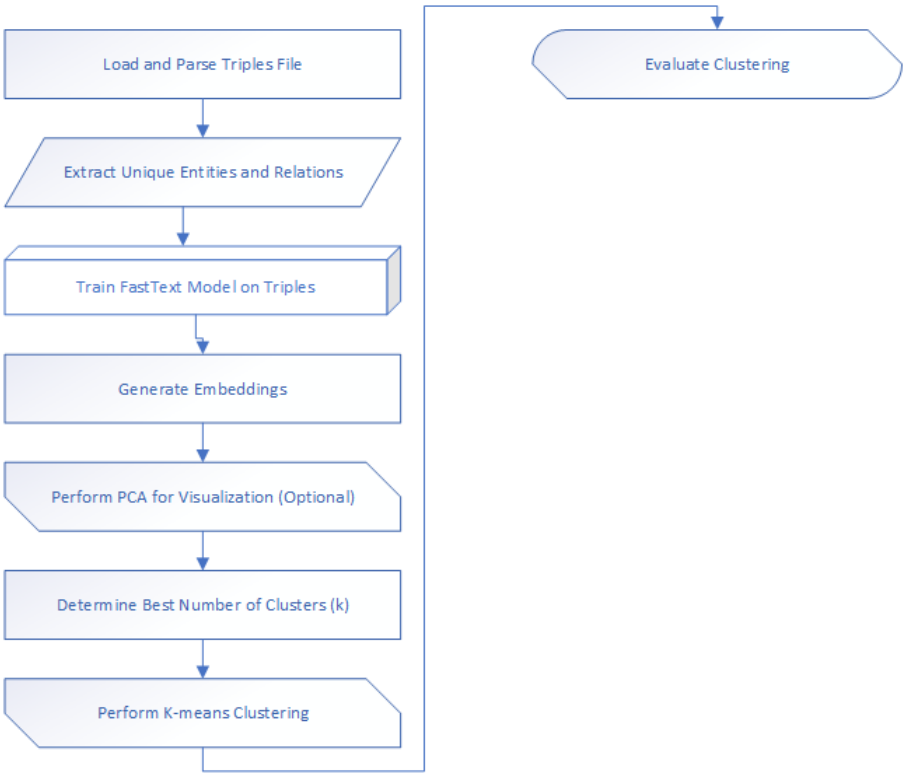


Figure 5.5: Clustering evaluation process

1.4 Models

Below, we outline the machine learning models used for the triple classification task. Two primary architectures are used. These architectures were derived after extensive experimentation, with a view to the efficiency and speed of model training. One architecture mostly utilizes dense layers, whereas the other predominantly employs 1D convolution layers. Subsequently, we use these architectures and add additional components to categorize our experiments based on the architecture and embeddings used, namely $TransE, TransH, HoIE, ComplEx, DistMult,$ and $RotatE$. Figure 5.6 depicts the initial architectural design. To prevent overfitting, a dropout layer follows each densely connected layer in the model architecture. At first, the architecture consists of a dense layer with 5012 units and a $ReLU$ activation function. In order to facilitate the model's acquisition of intricate patterns from the data, the activation function introduces nonlinearity. Following this layer, a dropout layer with a 0.5 rate randomly removes 50% of the nodes during training to avoid overfitting. The second layer—another dense layer with 2048 units and $ReLU$ activation—is complemented by a dropout layer with a 0.5 rate. This decrease in the number of units progressively reduces the dimensionality of the feature space.

The third dense layer then comprises 1024 $ReLU$-activated cells adjusted to a factor of 0.01 using L2 regularization. L2 regularization assists in mitigating overfitting by penalizing large weights. This dense layer is followed by a dropout layer with a rate of 0.5. The fourth dense layer has 512 units and also uses $ReLU$ activation and L2 regularization (0.01). It is followed by a dropout layer with a rate of 0.5, which keeps using dropouts for regularization. The fifth dense layer has 256 units with $ReLU$ activation and L2 regularization (0.01), followed by another dropout layer with a rate of 0.5 to make sure that the network is always regularized.

We flatten the result to produce a single long feature vector from these dense and dropout layers. This step is crucial for converting the multidimensional tensor output from the previous layers into a form suitable for the final dense layer. The final layer consists of a single unit and a tightly linked layer that utilizes a sigmoid activation algorithm. As it generates an output between 0 and 1, the sigmoid activation function is quite helpful for binary classification problems. This result shows the possibility of the positive class.

Figure 5.7 shows the second architecture. The architecture starts with $ReLU$ activation from a one-dimensional convolutional layer (Conv1D) with 128 filters and a 3-pixel kernel size. This layer incorporates L2 regularization to assist avoid overfitting by punishing significant weights and employs identical padding to guarantee the output has the same length as the input. The output of this convolutional layer is then sent to a batch normalization layer, which accelerates training, normalizes the activations and enhances stability. Following batch normalization, we randomly set a proportion of input units to zero during training to apply a dropout layer with a dropout rate of 0.5 and thus further reduce overfitting.

The second layer in the sequence is another Conv1D layer, but this time with 256 filters and the same kernel size of 3. It also uses $ReLU$ activation,$same$ padding, and L2 regularization. This layer, like the first convolutional block, follows batch normalization and a dropout layer with the same dropout rate, ensuring consistent regularization and normalization across the network.

After the convolutional layers, the model includes a flattening layer that converts the multi-dimensional tensor output from the previous layers into a single, long feature vector. This step is crucial for preparing the data for the fully connected, dense layers that follow. The first dense layer in this sequence has 512 units with $ReLU$ activation and includes L2 regularization. Another batch normalization layer and a dropout layer follow this layer, maintaining the pattern of normalization and dropout for regularization.

Finally, the model includes a dense layer with a single unit and sigmoid activation. This layer also includes L2 regularization to ensure the model's robustness. The sigmoid activation function is suitable for binary classification tasks, as it generates a probability value between 0 and 1 that denotes the probability of the positive class.

Subsequently, we may begin our experimentation. In addition to the embeddings, we also have two more components that are essential to our research. One layer is a graph convolution layer, while the other is a graph attention layer. We implement both layers using the TensorFlow framework. These layers undergo evolution throughout the tests. In this section, we provide a comprehensive analysis of the implementation and specific information about the trials conducted.

1.4.1 Group of Experiments

We classify the experiments based on the model design, the concatenation technique, and the additional components incorporated into the models. We were unable to perform all the trials for some models, mostly because of the extensive training period necessary. Here we provide a comprehensive list of both the group and individual tests conducted. All of the

models except those in Group 3 use Method 1 for negative sample generation. Technical difficulties or poor results prevented the completion of some experiments.

Figure 5.6: Dense architecture

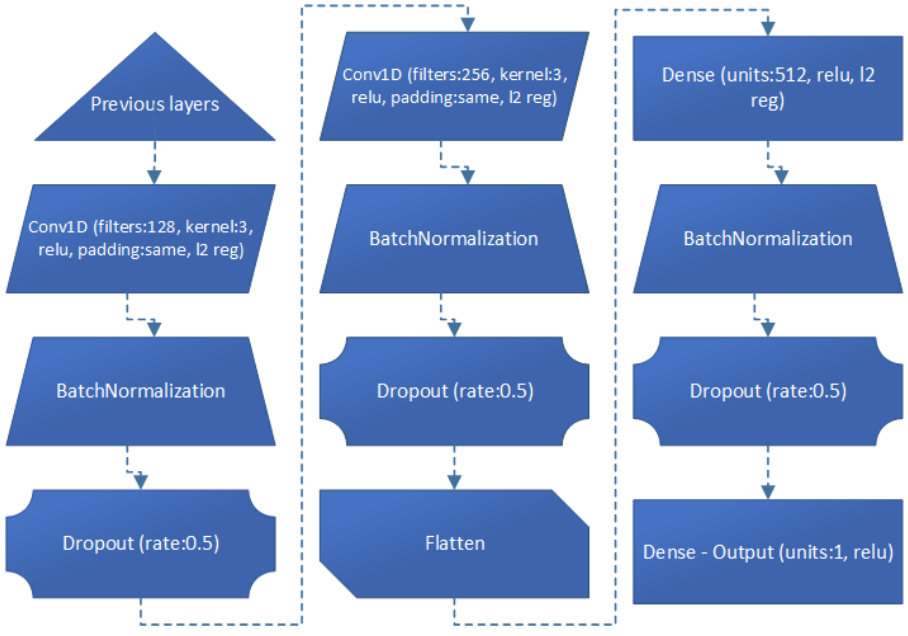


Figure 5.7: Conv1D architecture

Group 1

For this group we used the Dense architecture. Our six models, for this group, are shown in Figure 5.8 – Figure 5.13.

$TransE$: This model combines the Dense architecture with the $TransE$ algorithm, as seen in Figure 5.8. It adds embeddings for entity degree and inverse relation frequency to provide additional information to the model. Loading and preparing the knowledge graph data from a Turtle (TTL) file starts the process. The basis is graph parsing, entity mapping, and relation mapping to unique indices. This mapping facilitates the conversion of graph triples into a numerical format appropriate for neural network model input.

To ensure repeatability, the script sets the random seeds for TensorFlow, Python's random package, and NumPy to 42. Preprocessing generates relations and entities and converts triples to indexed forms. As shown in Method 1 above, we additionally construct negative instances to provide a balanced dataset including both positive and negative samples. Maintaining a 70/15/15 split ratio, we next shuffle and divide the dataset using Scikit-learn into training, validation, and test sets.

The model's core is a custom $TensorFlow\ Keras$ model that leverages entity and relation embeddings. Initially, $tf.keras.layers.Embedding$ generates these embeddings. A dense vector of a specified dimension, $embedding\_dim$, represents each entity and encapsulates its features. Similarly, we embed relations into vectors of the same dimensionality. During the training process, the model learns the embeddings for entities and relations, which enables it to capture latent patterns and interactions within the knowledge graph. To further enrich the entity representations, the model incorporates additional embeddings for in-degree, out-degree, and inverse relation frequency. We also create these embeddings using $tf.keras.layers.Embedding$. We create these embedding layers with a dimension of 1. The in-degree embedding captures the number of incoming edges to an entity, the out-degree embedding captures the number of outgoing edges, and the inverse relation frequency embedding captures the frequency of a relation in an inverse manner, emphasizing rare relations.

Using the input triples—subject, predicate, and object—the forward pass obtains the embeddings. For a given triple, we get the embeddings for the subject, predicate, and object. We improve the representations of the subject and the object by adding their in-degree embedding and out-degree embedding, respectively, thereby utilizing structural information from the graph. To change the representations of the subject and object embeddings depending on the frequency of the relation, we include the inverse relation frequency embedding in both of them. A batch normalization layer then normalizes the entity embeddings to ensure a constant scale and thus stabilize the training process. Following the $TransE$ model paradigm, the model computes the expected object embedding by merging the subject embedding and the relation embedding. To compute the final representation, the dense architecture concatenates the subject, relation, object, and expected object embeddings. These layers transform the concatenated embeddings into a single prediction value, which represents the likelihood of the supplied triple's validity within the knowledge graph.

Table 5.2 shows the choices we made about how to set up the optimizer, loss function, and training callbacks in the model to get the best performance and avoid overfitting. The Adam optimizer is selected with a learning rate of 0.00001. To suit the learning rate of any

parameter, Adam combines the benefits of two different modifications of stochastic gradient descent (SGD), namely Adaptive Gradient Algorithm ($AdaGrad$) and Root Mean Square Propagation ($RMSProp$). This makes it ideal for managing typical in knowledge graphs: noisy data and sparse gradients. We use the modest learning rate to provide fine-grained updates to the model parameters, which fosters steady convergence and prevents overshooting of the ideal values during training.

The model compilation uses binary cross-entropy loss. This loss function gauges the performance of a classification model by producing a probability value between 0 and 1. For this task, binary cross-entropy is suitable because it directly relates to the objective of estimating the probability of a given triple being legitimate. The reduction mode $SUM\_OVER\_BATCH\_SIZE$ averages the loss over the batch size, ensuring constant gradient updates regardless of the batch size.

To further improve the training process, the model uses a $ReduceLROnPlateau$ learning rate scheduler. If the validation loss does not improve for five consecutive epochs, this scheduler tracks it and lowers the learning rate by a factor of 0.6. This dynamic change enables the model to converge more effectively to a reduced validation loss by taking fewer steps as it approaches the ideal answer, thereby fine-tuning the learning rate during training. Set the minimal learning rate to 1e-7 to stop the learning rate from becoming too low and stopping advancement.

Additionally, an early stopping callback is employed to monitor the validation loss. If the validation loss does not improve for 5 consecutive epochs, training is stopped early, and the best model weights observed during training are restored. This prevents overfitting by stopping the training process before the model starts to learn noise in the training data, ensuring that the model maintains its ability to generalize to unseen data.

We evaluate the model's performance on the test set after training using various metrics like precision, *recall*, *F1 score*, *ROC AUC*, *PR AUC*, and *Matthew's correlation coefficient* (MCC). We also generate a confusion matrix to provide a detailed view of the model's classification performance. The model ends by showing the confusion matrix and the training versus validation loss plot. These show how the model learned and how well it could tell the difference between valid and invalid triples in the knowledge graph.

Table 5.2: Parameters for the TransE Model

Parameter	***Value***
Embedding Dimension	256
Number of Epochs	20
Batch Size	256
Optimizer	Adam
Learning Rate	0.00001
Loss Function	Binary Crossentropy

Metrics	Accuracy
Learning Rate Scheduler	ReduceLROnPlateau
Monitor	val_loss
Factor	0.6
Patience	5
Minimum Learning Rate	1e-7
Verbose	1
Monitor	val_loss
Patience	5
Restore Best Weights	True

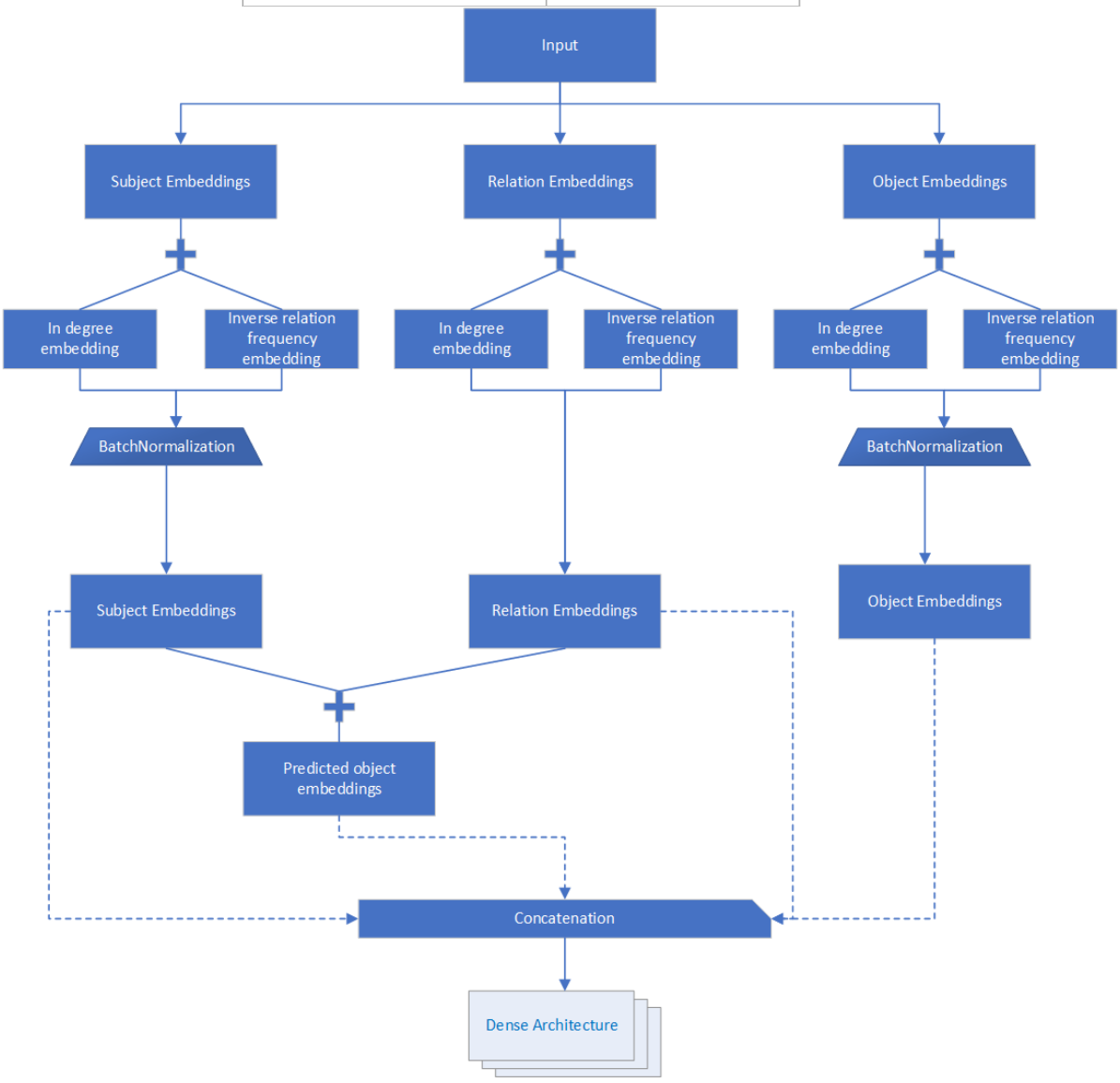


Figure 5.8: TransE Model

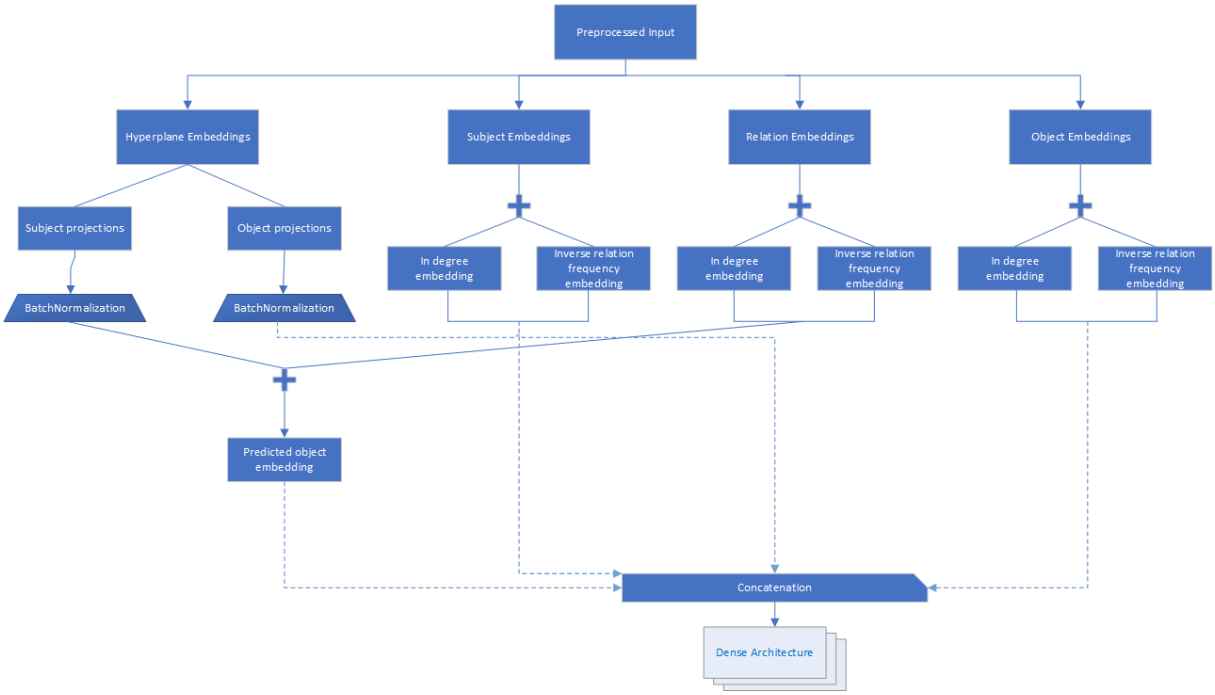


Figure 5.9: TransH Model

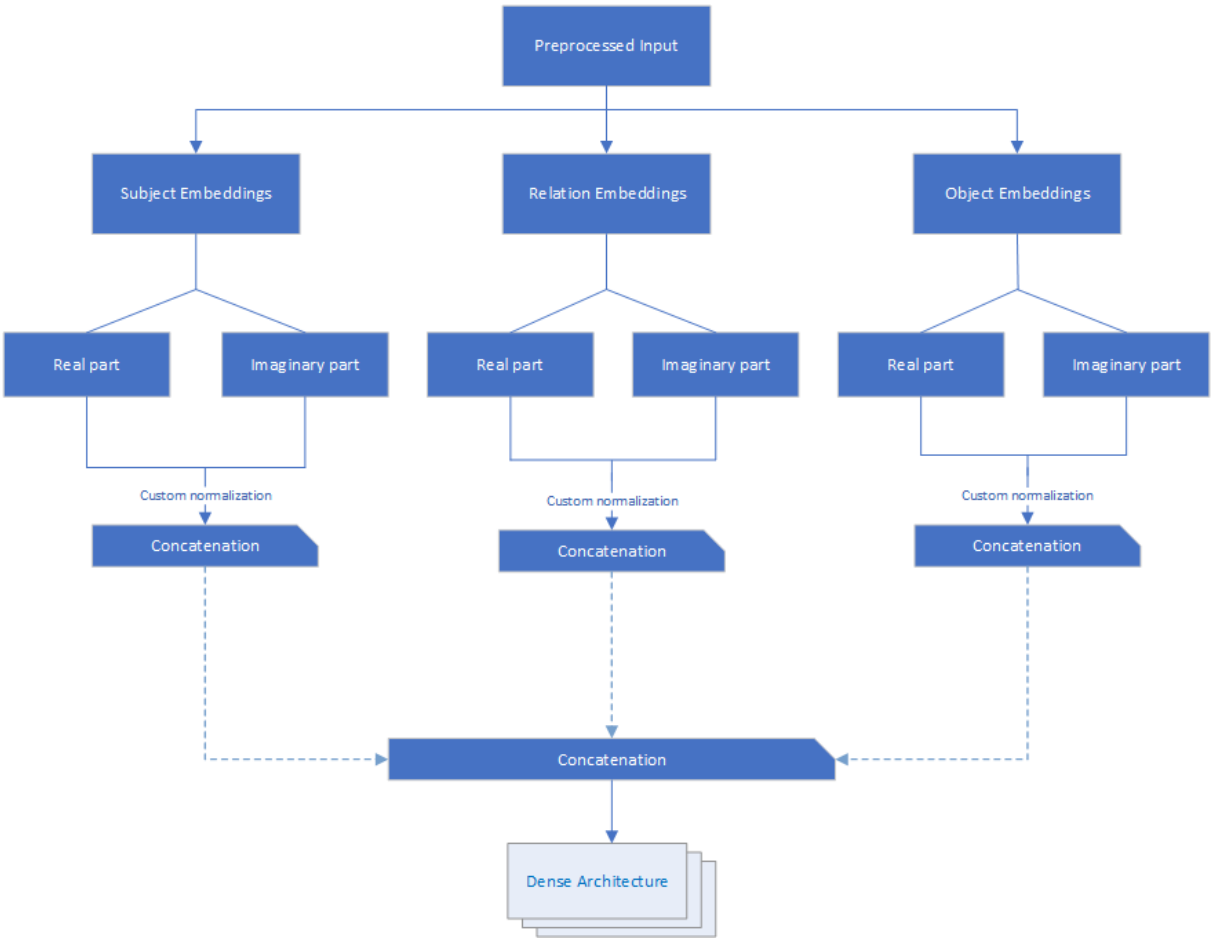


Figure 5.10:RotatE Model

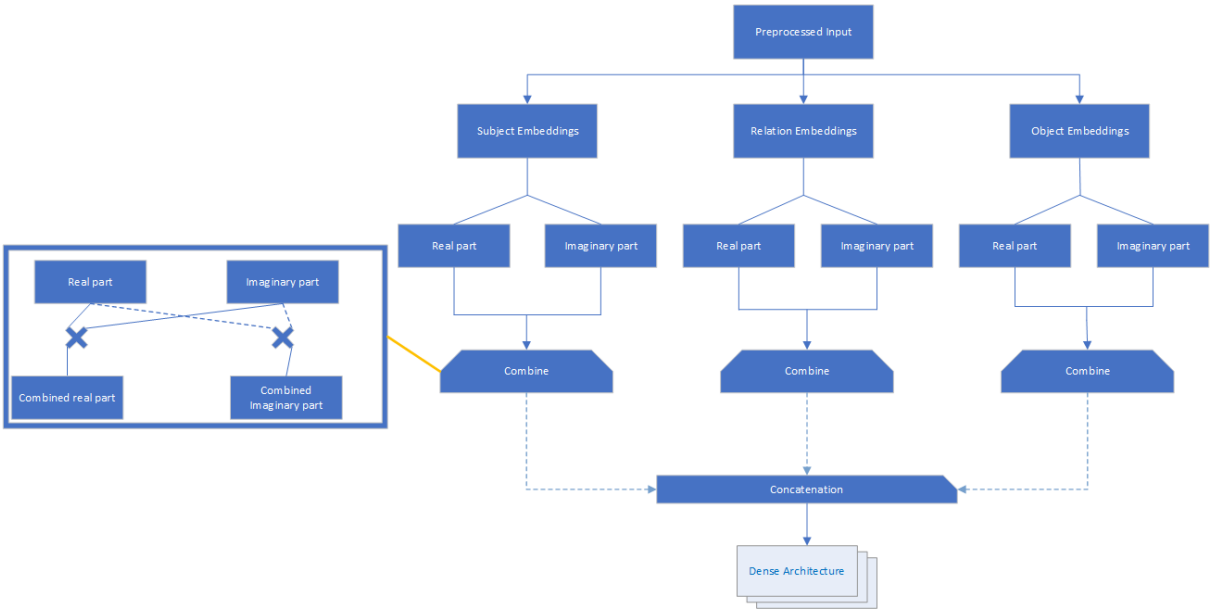


Figure 5.11: HoIE Model

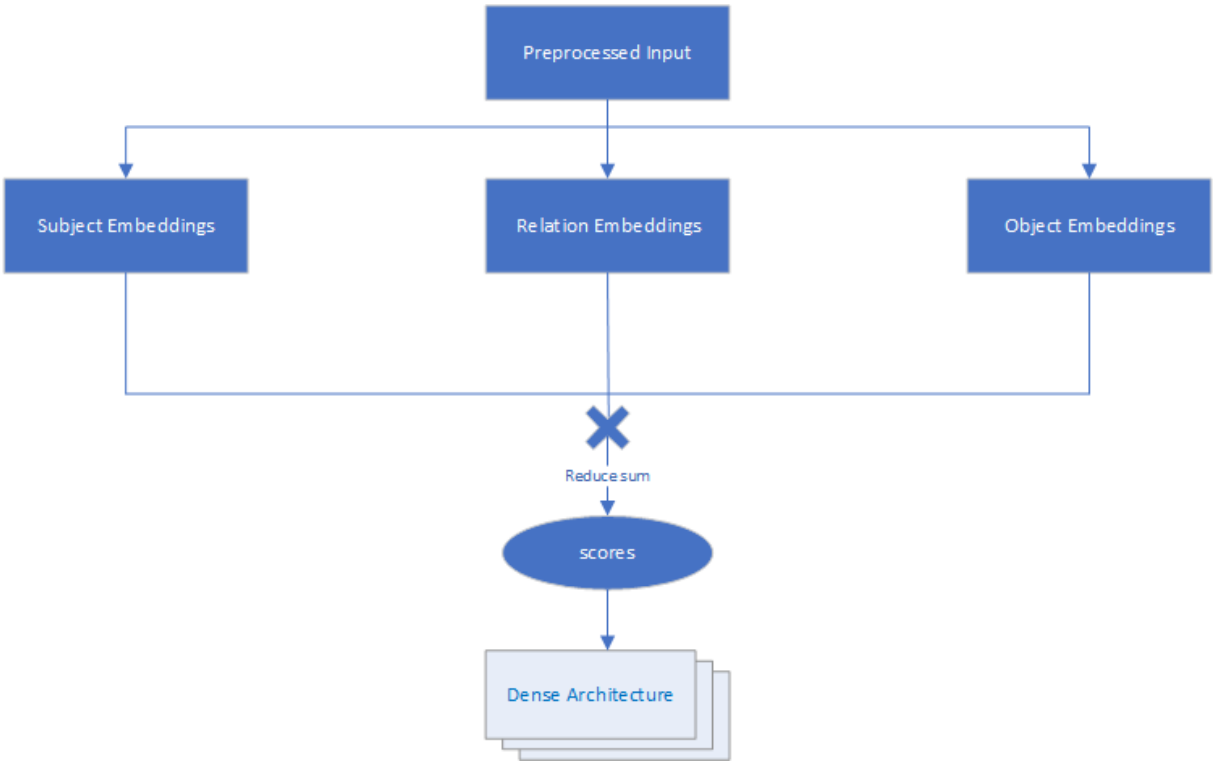


Figure 5.12: DistMult Model

$\underline{TransH}$*:* With the exception of the embeddings and the number of epochs parameter, which we set at 30, the architecture and parameters of this model are identical to those of the previous one. Because we use early stopping, the number of epochs doesn't play a critical role for most of the models. The model employs embeddings for entities and relationships, as well as additional embeddings for in-degree, out-degree, and inverse relation frequency, like in the $TransE$ model in this Group. It also incorporates dense layers for hyperplane projections and graph convolution techniques.

The $TransH$ approach associates each relation with a hyperplane and projects the entities onto these hyperplanes. We reshape the hyperplane embeddings and use them to perform matrix-vector multiplication with the entity embeddings, resulting in the projected entity embeddings. After creating a batch of normalization layers, we proceed to normalize these projected embeddings.

Next, we calculate a predicted object embedding by adding the relation embedding to the projected subject embedding. To form a combined embedding, the model concatenates the original subject embedding, the relation embedding, the projected object embedding, and the predicted object embedding. Figure 5.9 displays all the information mentioned above. The dense architecture processes this combined embedding.

We flatten the output from the final dense layer and pass it through a sigmoid-activated dense layer to produce a prediction. This prediction indicates the likelihood of the input triple being valid, allowing the model to perform triple classification tasks on the knowledge graph.

$\underline{RotatE}$: This approach uses real and imaginary components in complex embeddings for entities and relations. By rotating entity embeddings in the complex plane, the approach models relational patterns.

We then construct different embeddings for the imaginary and actual aspects of the relationships and entities. These embeddings reflect the subjects, predicates, and objects in the input triples. The technique requires normalizing these embeddings to maintain their magnitudes.

We compute their modulus and divide each component by it to normalize the entity embeddings—real and imaginary portions. We calculate the modulus by taking the square root of the sum of the squares of the real and imaginary embeddings. This procedure ensures that the embeddings coincide with the unit circle in the complex plane, which is very important for $RotatE's$ rotation.

The relationship has also been normalized. We then build combined embeddings for the subject, predicate, and object by concatenating the normalized real and imaginary components of the entity and relation embeddings.

We then concatenate these merged embeddings to provide a consistent triple representation. Figure 5.10 displays all the information mentioned above. We then run the concatenated embedding across the dense architecture. The use of dense layers simulates a simplified graph convolution process, enhancing the model's ability to learn from the graph structure. All the parameters are the same as in the above models.

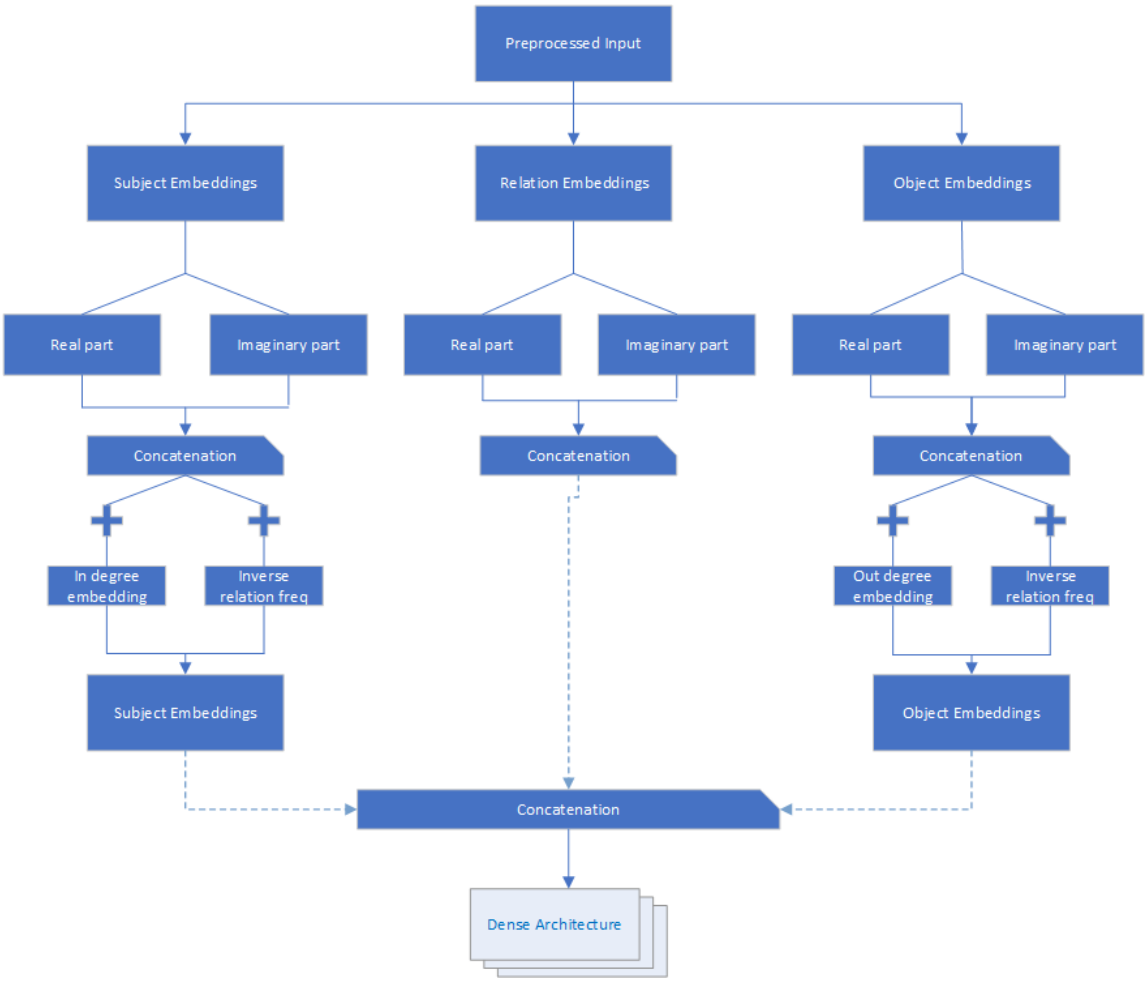


Figure 5.13: ComplEx Model

$\underline{HolE}$: The model is based on the Holographic Embeddings ($HoIE$) approach. It employs real-valued embeddings for entities and relations, as well as a combination function for embedding interactions.

Initially, we create embeddings for the real parts of entities and relations. The method uses the circular correlation of these embeddings to show how the entities and relations in the input triples interact with each other.

The function combines the real parts of the embeddings using element-wise multiplication (although circular correlation typically involves a more complex operation, this simplified version uses direct multiplication). This function concatenates the combined real part with itself to form the final embedding representation.

The model's call method fetches the real part embeddings for the subject, predicate, and object. The function then combines these embeddings to create embeddings for the subject, predicate, and object.

Next, we concatenate these combined embeddings to create a unified representation of the input triple. This concatenated embedding is passed through a series of dense layers, each of which is followed by a dropout for regularization.

We flatten the output from the final dense layer and pass it through a sigmoid-activated dense layer to produce a prediction. The remaining parameters remain unchanged from the models mentioned above. Figure 5.11 displays all the information mentioned above.

$\underline{DistMult}$: Here, we define the model using the $DistMult$ approach. This model employs embeddings for entities and relations and leverages element-wise multiplication to capture the interactions between these entities and relations.

Initially, we create embeddings for entities and relations, which represent the subjects, predicates, and objects in the input triples. The $DistMult$ approach involves calculating scores for the triples using element-wise multiplication of these embeddings.

The model's call method fetches the embeddings for the subject, predicate, and object. We compute the scores by performing an element-wise multiplication of the subject, predicate, and object embeddings, then summing the result across the embedding dimensions. This produces a score for each triple, indicating the strength of the relationship between the entities.

Next, we pass the computed scores to the Dense architecture. We flatten the output from the final dense layer and pass it through a sigmoid-activated dense layer to produce a prediction. The remaining parameters remain unchanged from the models mentioned above. Figure 5.12 displays all the information mentioned above.

$\underline{ComplEx}$: We define a model using the $ComplEx$ approach. This model employs real and imaginary embeddings for entities and relations, incorporating additional information such as in-degree, out-degree, and inverse relation frequency for each entity and relation.

Initially, we create embeddings for the real and imaginary parts of entities and relationships. For entities and relationships, we also use fixed embeddings to capture the in-degree, out-degree, and inverse relation frequencies. The difference between these and the above setups is that we set these embeddings to precomputed values, rendering them non-trainable. For this model, we calculate the in-degree, out-degree, and inverse relation frequencies in the preprocessing function as follows: These are arrays of zeros, each with a length equal to the number of entities, and another array, which has a length equal to the number of relations. We use these arrays to store the count of incoming edges for each entity, outgoing edges for each entity, and occurrences of each relation, respectively.

Next, the code iterates over each triple in the triples list, where is the subject, is the predicate (relation), and is the object. For each triple, it increments the out-degree of the subject entity (s) by 1, the in-degree of the object entity (o) by 1, and the frequency of the relation (e) by 1. This loop effectively counts how many times each entity appears as a subject or object, as well as how many times each relation appears in the triples. Finally, the code calculates the inverse relation frequency by taking the reciprocal of the relation frequency plus one. This addition of one ensures that there is no division by zero in cases where a relation has not appeared in the triples.

The model's call method retrieves the real and imaginary parts of the embeddings for subject, predicate, and object. We concatenate these embeddings to form the complete embeddings for each component of the triple.

We add the in-degree and out-degree embeddings to the subject and object embeddings, respectively. We add the inverse relation frequency embedding to both the subject and object embeddings to incorporate additional relational context. Next, we concatenate the combined embeddings for the subject, predicate, and object to create a unified representation of the input triple. We pass this concatenated embedding through the dense architecture. Figure 5.13 displays all the information mentioned above.

The model also includes skip connections implemented using linear layers. These skip connections connect the embeddings of the subject and predicate, as well as the object and predicate, to intermediate layers, allowing the model to capture more complex relationships. The implementation of skip connections necessitates the use of additional linear layers. These connections bypass specific layers in the dense network, allowing subject and predicate embeddings, as well as object and predicate embeddings, to be directly added to the intermediate layers of the network. More specifically:

- Skip Connection 1: This connection combines the subject and predicate embeddings. The first dense layer (dense1) adds these embeddings to its output. By incorporating this skip connection, the model preserves and directly feeds information about the subject and predicate into subsequent layers, thereby enhancing the learning process.
- Skip Connection 2: This connection combines the object's embeddings with the predicate. Similar to Skip Connection 1, it adds these embeddings to the output of the second dense layer (dense2). Again, this allows the model to retain information about the object, predict it, and incorporate it into the next layers of the network.

These skip connections serve to address the issue of vanishing gradients and facilitate the flow of gradient information during training. By providing direct paths for information flow, skip connections can help alleviate the problem of information loss in deep networks and enable more effective training.

We flatten the output from the final dense layer and pass it through a sigmoid-activated dense layer to produce a prediction.

Group 2

Here we use again the models from Group 1 with minor changes. The main change that we do in all models for this group is the use of the $StratifiedKFold$ method.

$\underline{TransE}$: Group 1's $TransE$ model takes a simple approach to data splitting. We first arbitrarily split the data into test, validation, and training sets. We divided the dataset into three categories: 15% for testing, 15% for validation, and 70% for training. We split this dataset only once to ensure a fixed division. This approach helps the model assess performance on the test set after training and tweak the training and validation sets, respectively. Although this method is computationally fast and simple to use, if the selected split does not reflect the general data distribution, it might cause either overfitting or underfitting. Furthermore, the lack of variety in the data splits suggests that the performance measures derived from this one test set may not be very adaptable for other unknown data.

By comparison, the $TransE$ model of this group makes use of stratified k-fold cross-validation, a more advanced and strong data splitting method. This approach divides the data into k equally sized folds—that is, k = 2. Every fold preserves the percentage of each class by having a balanced mix of positive and negative instances. We then run the model k times, each time using a separate fold as the validation set and the remaining k-1 folds as the training set. This procedure guarantees the use of every dataset instance for both training and validation, offering a comprehensive assessment of the model's performance across numerous data splits.

Stratified k-fold cross-validation has several benefits. Exposing the model to many subsets of the data during training and validation helps first reduce the danger of overfitting to a certain data division. By averaging the data from many folds, it also increases the dependability of the performance measures. This is a more realistic estimate of the model's generalizing capacity. Finally, this approach reveals possible variation in the model's performance across many data splits, providing information on its stability and resilience.

Nonetheless, stratified k-fold cross-valuation is computationally more demanding than a single train-test split. Multiple training and validation repetitions increase the computational overhead and the model's development time requirement. Still, the advantages of getting a more consistent and generally applicable assessment usually exceed the extra computational expenses. Furthermore, the embeddings changed. Although it does not explicitly start with precomputed values, the $TransE$ model of Group 1 contains inverse relation frequency embeddings. Conversely, we configure the inverse relation frequency embeddings to be non-trainable and start them with precomputed values from the dataset's relation frequencies.

This guarantees that, from the beginning, the embeddings fairly represent the actual distribution of relations in the graph. The in- and out-degree embeddings correspond to those in Group 1's model. We nonetheless include the entity degrees in the triples. First, we determine the degrees of the entities involved in the triples. These degrees serve as counts of the frequency with which every entity occurs in the dataset as a subject or objective. Method 1 computes the entity degrees, then produces negative triples. We then expand each negative triple to include the degrees of its subject and object entities.

We also extend positive triples based on the degrees of their individual components. We aggregate the enhanced positive and negative triples into a single dataset and generate matching labels to show whether every triple is positive (1) or negative (0). This approach, by considering how significant entities are in the graph, provides extra information to the triples (entity degrees), enabling the model to distinguish between valid and invalid triples. The Group 1 model's setup and parameters are also applicable here.

$\underline{TransH}$: For this model, we maintain the modifications we previously made for this group's $TransE$ model. We set the value of k to 3. We maintain the same parameters and setup as Group 1's model, except that the embedding dimension parameters are set to 512.

$\underline{RotatE}$: Again, we keep the changes we made above for this group's $TransE$ model. We set the value of k to 3. The only differences from Group 1's corresponding model are that we set the embedding dimension to 512 and concatenate each embedding once to the end, rather than separately for the subjects, relations, and objects.

$\underline{HoIE}$: Again, we keep the changes we made above for this group's $TransE$ model. We set the value of k to 3. We set the embedding dimension to 512, which differs from the corresponding model of Group 1. The rest of the parameters are the same as in the corresponding Group 1 model.

$\underline{DistMult}$: Again, we keep the changes we made above for this group's $TransE$ model. We set the value of k to 2. Because we use degree and inverse relation frequency embeddings, like in the $TransE$ model of this group, we don't directly pass the sum of the product of the embeddings to the dense architecture, like in the corresponding model of Group 1. First, we concatenate all the entity and relation embeddings, and then we pass the reduced sum of the product of the embeddings to the dense architecture.

$\underline{ComplEx}$: Again, we keep the changes we made above for this group's $TransE$ model. We set the k value to three. This model does not use skip connections, as the corresponding model in Group 1 does. We set the embedding dimension to 512. One last difference from the corresponding model of Group 1 is that we don't add the in, out, and inverse relation frequency embeddings to the corresponding subject, relation, and object embeddings, but we directly concatenate them to the subject, relation, and object embeddings.

Group 3

For this group of experiments, we use the Method 4 for negative samples generation.

$\underline{TransE, TransH}$: These models have the same setup and parameters as in Group 1.

$\underline{RotatE}$: The parameters of this model are the same as in the corresponding model of Group 1. However, there is a difference in the embeddings. To accommodate the model's requirements, we construct entity embeddings with double the usual dimensionality, dividing them into real and imaginary parts to form complex embeddings. This technique allows the model to utilize the geometric properties of complex numbers, which is crucial for modeling the rotational relationships inherent in certain types of data. The main entity embeddings are more accurate when they include extra embeddings that show the in-degree and out-degree of entities, as well as the inverse relation frequency. This is similar to what the $TransE$ model in Group 1 does.

This model applies uniform normalization across the entity embeddings using batch normalization, ensuring the standardization of embeddings to stabilize and improve the training process. The Group 1 model, on the other hand, normalizes the real and imaginary parts of the embeddings based on their modulus separately. It keeps the unit length of the embeddings, but it doesn't normalize them as well as this model does overall.

This model also includes a more complex operation that changes the embeddings into a complex space and uses the Fast Fourier transform (FFT) and its inverse (IFFT) to do circular correlation. This operation enables the model to compute relational patterns in a rotational manner, significantly enhancing the expressiveness of the embeddings. We then integrate the resultant into the dense layers for further processing, ensuring that these complex relational patterns contribute to the final prediction. The rest of the parameters are the same as the corresponding model in Group 1.

$\underline{HoIE}$: Here, we try to combine two different embedding methods. We use the exact setup in the embeddings, such as in this group's $TransE$ model. Then, in the dense architecture, before we make the final prediction, we use the method $combine\_hole\_embeddings$, the same as in the corresponding model in Group 1, to combine the subject, predicate, and object embeddings. We then concatenate the output of the flattening layer and the combined embeddings to make the final prediction. The rest of the parameters are the same as the corresponding model in Group 1.

$\underline{DistMult}$: In a similar vein, Group 1's $TransE$ model uses subject, predicate, object, in degree, out degree, and inverse relation frequency. After normalization, we compute the score, like in the corresponding model from Group 1, and concatenate this score with the subject, predicate, and object embeddings. The rest of the parameters are the same as the corresponding model in Group 1.

$\underline{ComplEx}$: This model is simpler than the one in Group 1. It doesn't use skip connections or structural information from the graph (degree and relation inverse frequency embeddings). It just uses real and imaginary embeddings for the subjects, predicates, and objects. After we concatenate these embeddings, we feed them into the dense architecture without normalizing them first. The rest of the parameters are the same as the corresponding model in Group 1.

Group 4

In this group of experiments, we introduce a graph convolution layer named $GCNLayer$, shown in Figure 5.14. We specifically design graph-based neural network models to perform convolution operations over graph-structured data. We initialize it with the number of units ($units$) and an optional activation function ($activation$). The units parameter specifies the output space's dimensionality, which determines the number of features in the output vectors for each node in the graph. The activation parameter enhances the expressive power of the layer's output by applying a non-linear activation function.

During the initialization phase, the $GCNLayer$ sets up its configuration based on the provided parameters. The call to the build method initializes the layer's learnable parameters. The build method specifically creates a weight matrix ($kernel$) with a shape defined by the input feature dimension and the specified number of output units. We initialize this weight matrix using the $Glorot$ uniform initializer, which sets the initial values to facilitate efficient training.

In the call method's forward pass, the layer applies the learned transformations to the input data in the forward pass of the call method. The matrix multiplication operation ($tf.matmul$) multiplies the input features by the weight matrix. This operation transforms the input features into the output space, which is defined by the number of units. The transformed features use the activation function, if specified during initialization, to introduce non-linearity into the model. The result is the $GCNLayer's$ final output, which can then be passed to subsequent layers or used for predictions.

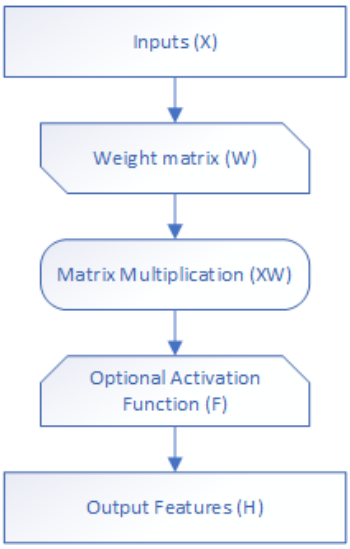


Figure 5.14: $GCNLayer$

$\underline{TransE}$: Here, we utilize subject, relation, object in and out degree, and inverse relation frequency embeddings, similar to those found in the corresponding model in Group 1. The model initializes extra embeddings, especially for $TransE$ representations of entities and relations, in order to use the $TransE$ method. Using $tf.keras.layers.Embedding$, the model first generates these embeddings—whose dimensionality matches that of the main embeddings. Including layers. These embeddings may capture distinct aspects of the relational structure, which the other model embeddings may not fully represent. This implementation directly integrates $TransE$-like embeddings into the model, unlike the traditional $TransE$ model, which combines embeddings using vector arithmetic to model relationships. We concatenate the embeddings for subject, predicate, and object with the outputs of the GCN layers; there are two $GCNLayers$ just before the dense architecture with parameters $units = embedding\_dim, activation = tf.nn.relu$. This direct concatenation allows the model to leverage the rich relational information embedded in the $TransE$-like vectors without explicitly performing the translational arithmetic.

The model effectively combines the relational features picked up by the embeddings with the structural features learned by the GCN by combining both regular and $TransE$-like embeddings with the outputs of the GCN layers. This hybrid approach allows the model to learn a more comprehensive representation of the graph data, capturing both local neighborhood structure and global relational context. Figure 5.15 displays all the above information. The model maintains the same parameters as Group 1's model, with the exception of setting the learning rate to 0.0001.

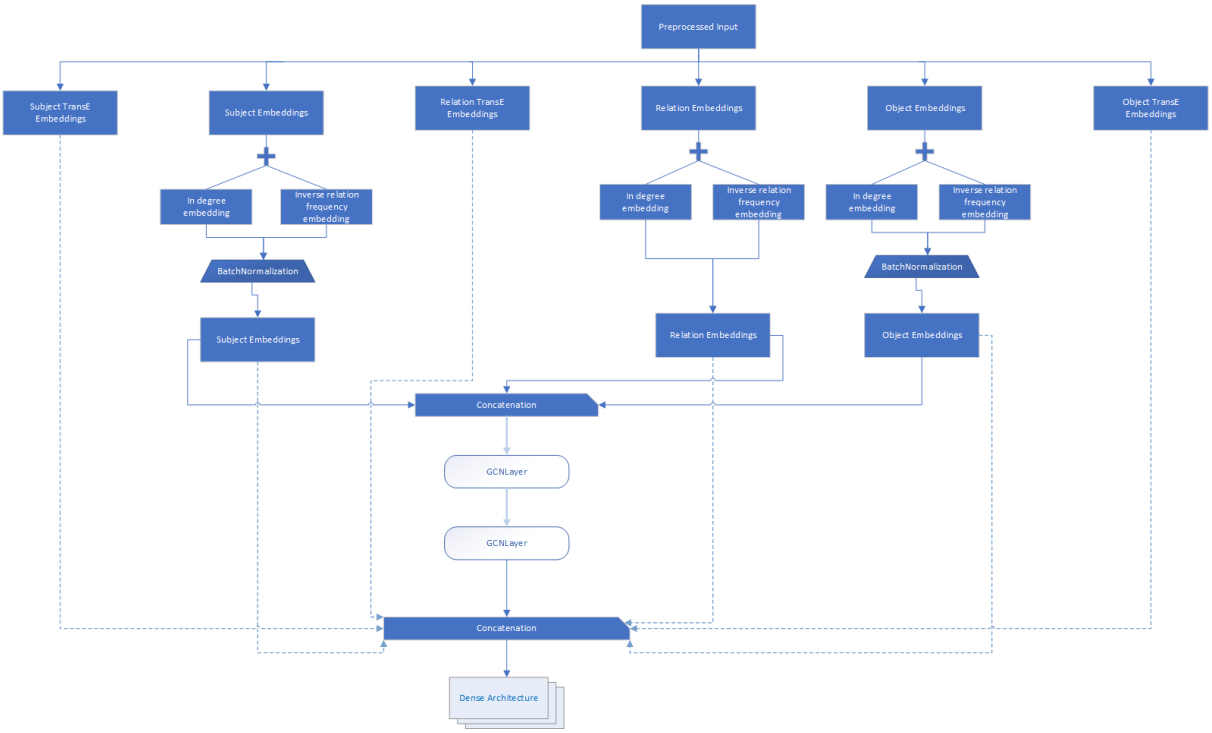


Figure 5.15: TransE Model for Group 4

$\underline{TransH}$: This model employs a different embedding strategy by combining a Graph Convolutional Network (GCN) and a more traditional approach from Group 1. Similar to the Group 1 model, it uses standard entity and relation embeddings, as well as scalar embeddings for in-degree, out-degree, and inverse relation frequency. However, instead of using hyperplane embeddings, this model employs a normal vector for each relation to define the hyperplane in the mechanism. We project entity embeddings onto these relation-specific hyperplanes by removing the entity embedding component along the normal vector. As a result, the entity embeddings project onto the hyperplane. The model then uses GCN layers to further refine these embeddings, exactly like in the $TransE$ model in this Group. The GCN layers gather data from nearby nodes and directly add relational context to the entity embeddings. We concatenate this aggregated information with the initial entity and relation embeddings, then pass it through the dense architecture for the final prediction. The model maintains the same parameters as Group 1's model, with the exception of setting the learning rate to 0.0001.

$\underline{HoIE}$: This model is based on $HoIE$ (Higher-order Interaction Embeddings) embeddings. It uses standard entity and relation embeddings and augments them with $HoIE$ embeddings, which encode higher-order interactions between entities and relations. We divided the $HoIE$ embeddings into two parts: one to augment the subject embedding, and another to augment the object embedding. This transformation captures more complex, higher-order interactions, which are crucial for understanding the multi-faceted nature of relationships in knowledge graphs. Additionally, this model includes embeddings for in-degree, out-degree, and inverse relation frequencies, like the $TransE$ model in Group 1. The model then normalizes these enriched embeddings and processes them through GCN layers, like in the $TransE$ model of this Group. We join the results of these GCN layers with the first embeddings to make a full feature vector that shows both local and global graph structures. We then pass this feature

vector through dense layers to make the final prediction. Group 1's parameters apply to the rest of the model, except for setting the learning rate to 0.0001.

$\underline{DistMult}$: Here, we use structural embeddings that are similar to those in Group 1's $TransE$ Model. Actually, the first part of the embeddings—until the normalization—is very similar to the $TransE$ model of Group 1. After normalizing the subject and object embeddings, we feed the subject, relation, and object embeddings into two GCN layers, like in this group's $TransE$ model. We concatenate them and feed them into the dense architecture. Then, in the dense architecture, after the flatten layer, we compute the reduced sum of the product of the flatten layer's output, the predicate, and the object embeddings. At the end, we make the prediction. The model maintains the same parameters as Group 1's model, except for setting the learning rate to 0.0001.

$\underline{ComplEx}$: This model is simpler than the corresponding one in Group 1. It doesn't use skip connections. It uses in-and-out degrees and inverse relation frequency embeddings, like in the $TransE$ model in Group 1. It uses real and imaginary parts for the subjects, predicates, and objects. Similar to the $TransE$ model in Group 1, we join the imaginary and real parts together. We achieve this by incorporating the in and out degrees and inverse relation frequency embeddings into the real part of the embedding. Next, we provide the concatenation of the subjects, predicates, and object embeddings as inputs to the two subsequent GCN layers, like in the $TransE$ model in this Group. The dense architecture then follows. The model maintains the same parameters as Group 1's model, with the exception of setting the learning rate to 0.0001.

$\underline{NoEmbds}$: This model uses embeddings, but it does so in the simplest possible form, without making any transformations to the embeddings. This model uses in, out, degree, and inverse relation frequency embeddings, as well as subjects, predicates, and object embeddings. It does the same things that the $TransE$ model in Group 1 did. Then it normalizes the embeddings again, like in the $TransE$ model in Group 1, and gives as input to the two following GCN layers the concatenation of the subject, predicate, and object embeddings. The dense architecture then follows. The model maintains the same parameters as Group 1's $TransE$ model, with the exception of setting the learning rate to 0.0001.

Group 5

In this group of experiments, we introduce a graph attention layer named $GATLayer$, shown in Figure 5.16. This layer, a sophisticated neural network layer, leverages the attention mechanism to perform graph-based operations. This layer is characterized by its ability to perform multi-head attention on the input features, enhancing the representation learning capability for graph-structured data. Several hyperparameters initialize the layer: the number of attention heads (num_heads), the dimensionality of the output space ($output\_dim$), the activation function, and the dropout rate ($dropout\_rate$).

In the initialization phase, the $GATLayer$ sets up the configuration for the attention heads and defines the activation function using the TensorFlow activation API. At this stage, we also define the dropout rate to prevent overfitting during training.

The call to the build method initializes the layer's learnable parameters. Specifically, it creates a weight matrix (kernel) and a bias vector (bias) for each attention head. The kernel matrices convert the input features into the output space, adding biases to these transformed features. During initialization, we specify both the input feature dimension and the output dimension. To ensure proper weight initialization, we use the $Glorot$ uniform initializer for the kernels and initialize the biases to zeros.

During the forward pass, implemented in the call method, the layer applies the learned transformations to the input data. The layer multiplies each attention head's input features by the corresponding kernel matrix, then adds the corresponding bias. These operations generate multiple transformed feature sets, one for each attention head. We then concatenate the outputs from all heads along the last dimension, effectively combining the multi-head attention results. We pass the concatenated outputs through the specified activation function to introduce non-linearity, thereby enhancing the expressive power of the layer. During training, we apply dropout to the activated outputs to improve generalization, randomly omitting certain neurons with a probability defined by the dropout rate.

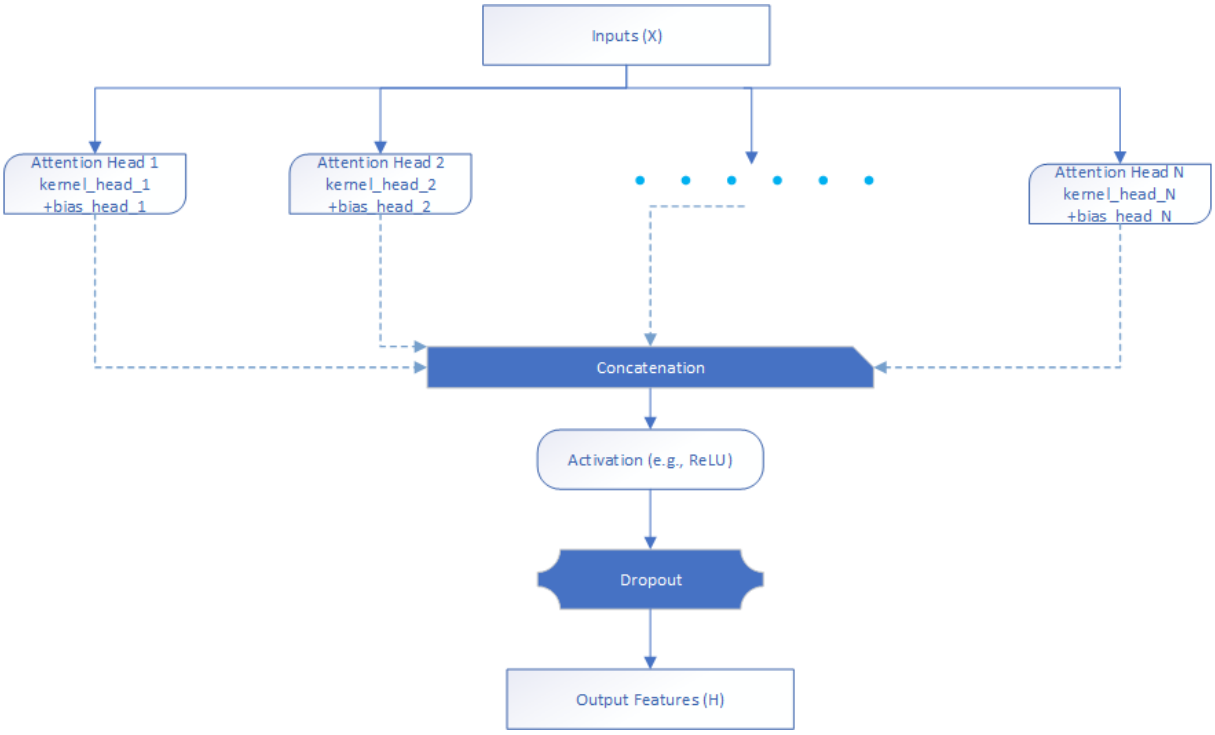


Figure 5.16: GATLayer

For this set of experiments, for the most part, we modify the dense architecture of Figure 5.6 slightly to simplify it. We remove all the dropout layers and the flatten layer from the model, and we apply L2 regularization to all dense layers. We evaluate and utilize the models on the same graph, so allowing the model to slightly overfit during training may not necessarily be detrimental. This is because the model's overfitting implies that it has learned to accurately capture the specific patterns and relationships within the training graph. Therefore, the model's high performance on the training data should theoretically transfer to the test data in the same graph, as it is unlikely to encounter significantly different structures or relationships. However, it is important to acknowledge the potential risks associated with

overfitting. Most of the models in this Group use the $MarginRankingLoss$, which has a margin of 0.1.

$\underline{TransE}$: For this model, the embeddings are pretty straightforward. We use subject, predicate, and object embeddings. We perform the transformation for object embeddings, similar to the corresponding model in Group 1, then concatenate all the embeddings, apply a dropout (rate 0.5), and then apply one $GATLayer$ ($num\_heads = 4, output\_dim = embedding\_dim, dropout\_rate = 0.5$). Then the dense architecture follows. We also use a different loss function, $MarginRankingLoss$ (margin 0.1), which we use not only here but for many experiments below. Machine learning uses this custom loss function to rank positive samples higher than negative samples by a specified margin.

The $MarginRankingLoss$ class, shown in Figure 5.17, is a subclass of $tf.keras.losses$. TensorFlow-based neural network models can seamlessly integrate with Loss. This class initializes with a single parameter, margin, that determines the margin for ranking positive samples higher than negative samples. The application's specific requirements can adjust the default value of 0.1 for this margin.

In the call method, the core functionality of the margin ranking loss is defined. This method computes the loss based on the predicted scores (y_pred) and the true labels (y_true). We assume the true labels to be binary, where 1 signifies positive samples and 0 indicates negative ones. The process begins by separating the predicted scores into positive and negative scores using $tf.boolean\_mask$, which filters the predictions based on the true labels.

To make it easier to compute pairwise differences, we reshape the isolated positive and negative scores. We expand positive scores along the rows and negative scores along the columns. This reshaping allows the creation of a margin matrix, where each element represents the difference between a positive and a negative score, adjusted by the specified margin.

We compute the margin matrix by subtracting the positive and negative scores from the specified margin. We then pass this matrix through a $tf.maximum$ function, which effectively applies the hinge loss concept by ensuring that only positive differences contribute to the loss. Using $tf.reduce\_sum$, we calculate the final loss by summing all positive values in the margin matrix.

This loss function encourages the model to rank positive samples higher than negative ones by at least the specified margin, thus promoting a clear separation between positive and negative predictions. The model maintains the same parameters as Group 1's $TransE$ model, with the exception of setting the learning rate to 0.0001 and the embedding dimension to 512.

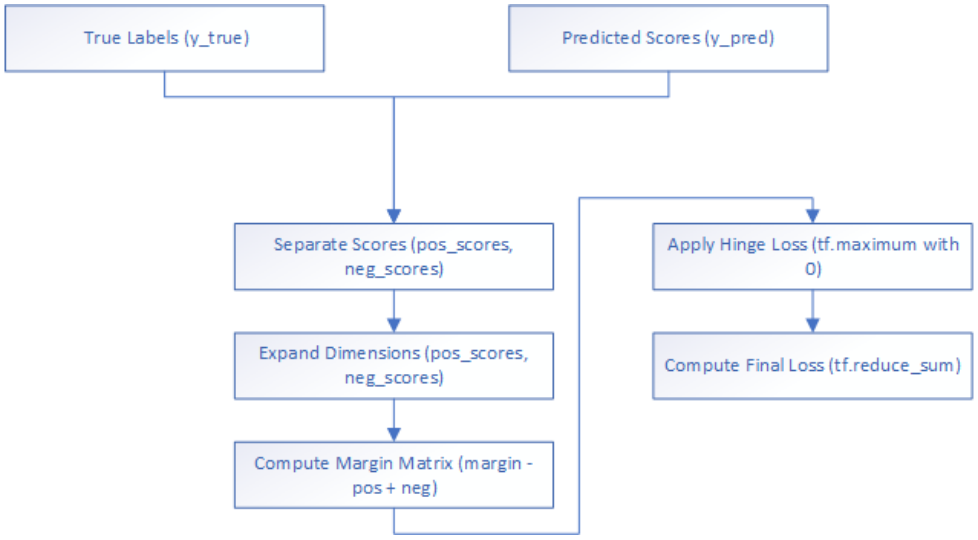


Figure 5.17: MarginRankingLoss function

$\underline{TransH}$: Group 4's corresponding model employs a simple embedding strategy, initializing entity and relation embeddings using standard embedding layers. We create the entity embeddings using $tf.keras.layers.Embedding$. We create the embedding using parameters for both the number of entities and the embedding dimension. Similarly, we create relation embeddings based on the number of relations. Additionally, another embedding layer creates a normal vector specific to each relation, representing $TransH$-like embeddings. We use this normal vector to project entity embeddings onto relation-specific hyperplanes, helping to model the multi-faceted nature of relationships in the data. We further refine the embeddings by incorporating degree and relationship information.

In contrast, this model doesn't use degree and relation information and adopts a more complex approach by distinguishing between embeddings in two different spaces: the entity space and the hyperplane space. The model specifically creates separate entity and relation embeddings for the entity space and the hyperplane space. By handling projections and transformations differently in each space, this dual embedding approach enables more sophisticated modeling of the interactions between entities and relations. This model projects entity embeddings onto relation-specific hyperplanes in a way that is similar to Group 4's model, but with the added complexity of embeddings having their own spaces. To avoid overfitting, we concatenate the projected embeddings with the relation embeddings, then process them through a dropout layer. Then follows one GATLayer, like in the $TransE$ model in this Group, and the dense architecture. The model maintains the same parameters as Group 5's $TransE$ model.

$\underline{RotatE}$: This model utilizes both entity and relation embeddings. Using $tf.keras.layers.Embedding$, we initialize the entity embeddings. To accommodate complex-valued representations, the embedding layer has a dimensionality twice that of the embedding layer. Similarly, we initialize the relation embeddings with the same dimensionality. These embeddings are critical for representing the entities and relations within a complex vector space, which enhances the model's ability to capture intricate relational patterns. This model uses a custom layer the $RotatEEncoder$, which we use it to encode the entity embeddings.

The $RotatEEncoder$, a custom layer that encodes entity embeddings through a rotation-based transformation, is a unique aspect of this model. The $RotatEEncoder$ takes advantage of complex numbers to $RotatE$ entity embeddings in a way that reflects the relational structure. This encoding process is essential for modeling the interactions between entities and relations effectively. The encoder splits the input embeddings into real and imaginary parts, then applies rotational transformations using learned weights. Specifically, it multiplies the real part of the embeddings by the cosine of the learned weights and subtracts the imaginary part multiplied by the sine of the weights to obtain the rotated real part. On the other hand, we obtain the rotated imaginary part by multiplying the real part by the sine of the weights and then adding the multiplied imaginary part by the cosine of the weights. The encoded entity embeddings use the concatenated representation of the rotated real and imaginary parts.

After encoding the entity embeddings, the model calculates the circular correlation between the entity and relation embeddings. The model performs complex multiplication by multiplying and summing the real and imaginary parts of the embeddings, which reflects the intricate interactions between entities and their relations. The resulting complex correlation captures relational semantics in a way that simple real-valued operations cannot.

To prevent overfitting and enhance generalization, the model incorporates a dropout layer with a specified dropout rate, which stochastically sets input units to zero during training. The model applies this dropout to the complex multiplication's output, thereby introducing regularization. Using one GATLayer, like in the $TransE$ model in this Group, the model further refines the embeddings. Then the dense architecture follows. The model maintains the same parameters as Group 5's $TransE$ model, with the exception that it uses the $BinaryCrossentropy$ loss function.

$\underline{HoIE}$: This model uses both real and imaginary components for entity and relation embeddings. It combines them to make the most of the $ComplEx$ embedding model's features, which handle complex relational patterns well. For the real and imaginary components, we initialize the entity and object embeddings using separate embedding layers. To enhance the representational capacity, the model incorporates higher-order interaction embeddings ($HoIE$). An embedding layer, $hoie\_embeddings$, creates these by encoding additional relational information for each entity. We divide the $HoIE$ embeddings into real and imaginary components, doubling the embedding dimension to capture more nuanced interactions. A holographic embedding function, $combine\_hole\_embeddings$, merges these components by multiplying the real and imaginary parts and concatenating them. We apply this process to both subject and object embeddings, yielding s_hoie and o_hoie, respectively.

The forward pass concatenates the real and imaginary parts of the subject, predicate, and object embeddings to form the full complex embeddings, s_embed, p_embed, and o_embed. These complex embeddings then merge with the holographic embeddings. Then follows one GATLayer, like in the $TransE$ model in this Group, and the dense architecture; for this experiment, we use the full architecture with the dropout layers. The remaining layers and model parameters are identical to the other models in this group.

$\underline{DistMult}$: For this model, we use subject, predicate, and object embeddings, and the product of these embeddings is used as a score. Then we apply a dropout layer. Then follows one GATLayer, like in the $TransE$ model in this Group, and the dense architecture. This model uses the same setup and parameters as the $TransE$ model in Group 5.

$\underline{ComplEx}$: For this model, we use real and imaginary parts for entity and relation embeddings. We initiate the embeddings using separate embedding layers for the real and imaginary parts of both entities and relations. Additionally, the model includes embeddings for entity degrees and inverse relation frequencies, similar to the $TransE$ model in Group 1. The forward pass retrieves the real and imaginary parts of the subject, predicate, and object embeddings. We then concatenate these embeddings to create the combined $ComplEx$ embeddings. We then further combine these concatenated embeddings into a single tensor, $concat\_embed$, which encapsulates the information. A GATLayer, like in the $TransE$ model in this Group, processes the tensor. The dense architecture then follows. This model uses the same setup and parameters as the $ComplEx$ model in Group 4.

Group 6

For this group, we use both $GCNLayer$ and $GATLayer$ in the models. Also, for most experiments, we use the dense architecture without dropout layers and with l2 regularization at all layers, as in Group 5. Most of the models in this Group use the $MarginRankingLoss$ with a margin of 0.1.

$\underline{TransE}$: Here, we use subject, predicate, and object embeddings, similar to the $TransE$ model in Group 1. After the concatenation, we apply a dropout layer (rate 0.5), followed by two $GCNLayers$ with parameters $units = embedding\_dim, activation = 'relu'$, and $GATLayers$ with parameters $num\_heads = 4, output\_dim = embedding\_dim, dropout\_rate = 0.5$. The dense architecture then follows. Group 5 contains the remaining parameters and configuration of the $TransE$ model.

$\underline{TransH}$: Here, we implement the model by projecting the entity embeddings onto the relation hyperplanes. We achieve this projection by subtracting the entity embeddings' parallel component to the relation embedding, which effectively positions the entity embeddings onto the relation embedding's defined hyperplane. Additionally, the relation projection embeddings transform, normalize, and reshape the relation embeddings appropriately.

Following the projections, we concatenate the projected subject, predicate, and object embeddings into a single representation. Then follows a dropout layer, with a rate of 0.5, two GCN and GAT layers, like in the $TransE$ model in this Group, and the dense architecture. The rest of the parameters and configuration are the same as in the corresponding model in Group 5.

$\underline{RotatE}$: For this model, we make some changes to the $MarginRankingLoss$, the GAT, and the GCN layers. The $MarginRankingLoss$ computes the loss by comparing positive and negative scores directly, whereas the $ComplexMarginRankingLoss$ separates the real and imaginary parts of the embeddings, handling them independently before combining the losses. Both the $GATLayer$ and the $GCNLayer$ work with real-valued inputs. The $GATLayer$

uses attention mechanisms to focus on important parts of the graph, while the $GCNLayer$ gathers data from nodes that are close by. On the other hand, the $ComplexGATLayer$ and $ComplexGCNLayer$ cater to complex-valued inputs, separating the real and imaginary components for independent processing before merging them.

Also, this model uses a more complex encoder-decoder mechanism based on complex-valued operations. The $RotatEEncoder$ in this model creates separate real and imaginary embeddings for entities and relations. It uses these embeddings to produce complex-valued representations of the subject (s_embed), predicate (p_embed), and object (o_embed). The $RotatEDecoder$ then computes the score by reducing the sum of the element-wise product of the complex embeddings. In the model, we use real and imaginary embeddings; we concatenate the real and imaginary embeddings; we transform the object embedding, calculating the product of subject and predicate embeddings; and we concatenate the subject, predicate, and transformed object embeddings. The model then follows two GCN and GAT layers, like in the $TransE$ model in this Group. The dense architecture then follows. The rest of the parameters and configuration are the same as in the corresponding model in Group 5.

$\underline{HoIE}$: This model retrieves the corresponding real and imaginary embeddings for the subject and object entities, as well as the predicate relation. We then combine the embeddings to create relation-specific transformed embeddings for the object, which we calculate as follows: We derive the real part of the object embedding by subtracting the product of the subject's real and predicate's real embeddings from the product of the subject's imaginary and predicate's imaginary embeddings. We obtain the imaginary part by adding the product of the subject's real and predicate's imaginary embeddings to the product of the subject's imaginary and predicate's real embeddings.

Next, the model concatenates the real and imaginary parts of the subject, transformed objects, and original object embeddings, resulting in a complex embedding that integrates both real and imaginary components. A dropout layer with a rate of 0.5 further processes this combined embedding to prevent overfitting during training. Then follow two GCN and GAT layers, like in the $TransE$ model in this Group, and the dense architecture. The rest of the parameters and configuration are the same as in the corresponding model in Group 5.

$\underline{DistMult}$: For each triple, the model retrieves the corresponding embeddings for the subject, predicate, and object, like in most of the other models. The model then subjects these embeddings to a dropout layer with a rate of 0.5, randomly setting a fraction of the embedding values to zero during training, thereby preventing overfitting and enhancing generalization. Following dropout, the embeddings pass through two GCN and GAT layers, like in this Group's $TransE$ model. The rest of the parameters and configuration are the same as in the corresponding model in Group 5.

$\underline{ComplEx}$: For this model, we use real and imaginary parts for subject, predicate, and object embeddings. The model then performs a relation-specific transformation on the object embeddings. In this transformation, we perform complex multiplication, multiplying the real part of the subject embedding by the real part of the predicate embedding and subtracting it from the product of the imaginary parts. The real part of the predicate embedding multiplies the imaginary part of the subject embedding simultaneously, adding it to the product of the

real part of the subject embedding and the imaginary part of the predicate embedding. This results in the transformed real and imaginary parts of the object embedding.

Next, to form a combined embedding, the model concatenates the subject's real and imaginary parts, transformed objects, and object embeddings. We concatenate the real parts together, and similarly, we concatenate the imaginary parts together. We then combine these concatenated real and imaginary embeddings to form the final complex embedding. To avoid overfitting, we subject the concatenated embeddings to a 0.5 dropout rate. Next, the model incorporates two GCN and GAT layers along with a dense architecture, mirroring the other models in this group. The rest of the parameters and configuration are the same as in the corresponding model in Group 5.

Group 7

For this group of experiments, we use the Conv1D architecture (Figure 5.7) combined with the different embedding manipulation methods we use above.

$\underline{TransE}$: Here, we utilize subject, predicate, and object embeddings, similar to those found in the $TransE$ model in Group 1. In Table 5.3, we can see the parameters of this model. Most models that use the Conv1D architecture will use these parameters.

Table 5.3: Parameters for the TransE model in Group 7

Parameter	***Value***
Embedding Dimension	400
Number of Epochs	15
Batch Size	256
Optimizer	Adam
Learning Rate	0.00005
Loss Function	Binary Crossentropy
Metrics	Accuracy
Learning Rate Scheduler	ReduceLROnPlateau
Monitor	val_loss
Factor	0.5
Patience	4
Minimum Learning Rate	1e-7
Verbose	1
Monitor	val_loss
Patience	4
Restore Best Weights	True

$\underline{TransH}$: The model embeds entities and relationships, resulting in dense vector representations for each entity and relation. The relation projections define a hyperplane onto which we project the subject embeddings. This projection operation involves subtracting the dot product of the subject embedding and the relation projection from the subject embedding itself, scaled by the relation projection. We then translate the projected subject embedding by adding the relation embedding, resulting in the transformed object embedding. Additionally, we compute a $TransE$-like embedding by simply adding the relation embedding to the subject embedding, forming the predicted object embedding.

To form a comprehensive embedding tensor, we concatenate these transformed and predicted embeddings with the original subject and relation embeddings. To prepare for the Conv1D architecture, we expand this concatenated tensor with an additional channel dimension. This model sets patience to 2 in $ReduceLROnPlateau$. The remaining parameters and configuration and setup are identical to those of the $TransE$ model in this Group.

$\underline{RotatE}$: Initially, we embed entities and relations using $tf.keras.layers.Embedding$. The process of embedding involves representing each entity with a vector twice the length of the specified embedding dimension, and each relation with a vector of the same embedding dimension. In addition to these standard embeddings, the model uses separate embeddings

for the $RotatE$ approach and stores them in embedding layers with dimensions scaled appropriately for complex number operations, just like before.

For each input triple, we retrieve the corresponding embeddings for both the standard and $RotatE$ embeddings. We then split the $RotatE$ embeddings into real and imaginary parts, allowing us to perform complex number operations. We specifically split the subject and object embeddings into their respective real and imaginary components.

The $RotatE$ approach's core involves applying a rotation to the embeddings. We achieve this by separately calculating the real and imaginary components of the score. We obtain the real score by summing the product of the relation embedding, the product of the real parts, and the product of the imaginary parts of the subject and object embeddings. We compute the imaginary score similarly, subtracting the product of the real part of the subject and the imaginary part of the object from the product of the imaginary part of the subject and the real part of the object. We then stack and sum these real and imaginary scores to generate a final score. We expand this final score with an additional channel dimension to prepare it for further processing by the Conv1D architecture. The remaining parameters and setup are identical to those in this group's $TransE$ model.

$\underline{NoEmbds}$: For this model, we simply use subject, predicate, and object embeddings, concatenate them, and feed the result to the Conv1D architecture. The remaining parameters and setup are identical to those of the $TransE$ model in this Group.

$\underline{HoIE}$: Initially, entities and relations are embedded using $tf.keras.layers.Embedding$. Additionally, an embedding layer stores $HolE$ embeddings with dimensions scaled appropriately for real and imaginary parts.

For each input triple, the system retrieves both the standard and $HolE$ embeddings. We compute the $HolE$ embeddings by combining the real and imaginary parts of the embeddings using a circular correlation operation, similar to the $HoIE$ model in Group 1. We then pass the resulting $HolE$ embeddings through two dense layers with $ReLU$ activation functions to ensure non-linearity. These dense layers serve to capture complex interactions between entities, enhancing the expressiveness of the model.

The modified concatenation process combines the computed $HolE$ embeddings with the input embeddings. The Conv1D architecture then passes the concatenated embeddings, which include subject, predicate, and object embeddings, along with the computed $HolE$ embeddings for both subject and object. These convolutional layers apply filters to capture patterns in the input embeddings, aiding in feature extraction. The remaining parameters and setup are identical to those of the $TransE$ model in this Group.

$\underline{DistMult}$: Here, exactly like the corresponding model in Group 1, we compute the $distmult$ score of the embeddings and pass it through the Conv1D architecture. The remaining parameters and setup are identical to those of the $TransE$ model in this Group.

$\underline{ComplEx}$: For this model we use real and imaginary parts for embeddings only for the subject and predicate. Then we concatenate these embeddings and feed them into the Conv1D architecture. The remaining parameters and setup are identical to those of the $TransE$ model in this Group.

Group 8

For this Group we use the Conv1D architecture combined with the models in Group 7 and the $GCNLayer$.

$\underline{TransE}$: For this model, each entity and relation in the input triples is represented using two types of embeddings: $TransE$ embeddings and standard embeddings. These embeddings have the same dimensions and are retrieved from their respective $tf.keras.layers.Embedding$ layers, with $TransE$ embeddings capturing translational relationships and standard embeddings providing more general representations. These embeddings are then concatenated to form a comprehensive embedding vector that encapsulates both types of representations. Specifically, the $TransE$ embeddings of the subject, predicate, and object are concatenated together, followed by concatenation with the standard embeddings. The combined embeddings are then processed through the Conv1D architecture. Two $GCNLayers$ are added to the Conv1D architecture after the second dropout layer in the architecture and before the flatten layer, with parameters $units = 256, activation = 'relu'$ and $units = 256, activation = 'relu'$ correspondingly. The remaining parameters and setup are identical to those of the $TransE$ model in Group 7.

$\underline{TransH}$: This model is the same as the corresponding model in Group 7, adding the $GCNLayers$ as in the $TransE$ model of this Group. The remaining parameters and setup are identical to those of the $TransE$ model in this Group.

$\underline{RotatE}$: For this model, we introduce an improved version of the $GCNLayer$ shown in Figure 5.18: Improved GCNLayer. The first $GCNLayer$ is a simple implementation of a graph convolutional layer. It initializes with a specified number of units and an optional activation function. The layer initializes its weights (kernel) and biases using $Glorot$ uniform and zero initializers, respectively. During the forward pass, the kernel multiplies the input and adds a bias. If provided, the output will receive the activation function.

The second $GCNLayer$ introduces several enhancements over the first version. It allows for an optional bias term and kernel regularization, providing greater flexibility and control over the layer's behavior. The use_bias parameter allows for bias inclusion or exclusion. Any specified regularizer can regularize the kernel, adding a layer of robustness and preventing overfitting. This version also incorporates a residual connection, which adds the original input to the output following the kernel transformation and activation. This link helps train deeper networks by reducing the vanishing gradient issue. This makes the gradient flow better during backpropagation. Furthermore, the kernel initialization is explicitly defined using the $GlorotUniform$ initializer.

This model is exactly the same as in the corresponding model in Group 7, adding two improved $GCNLayers$ in the same way as in the $TransE$ model in this Group.

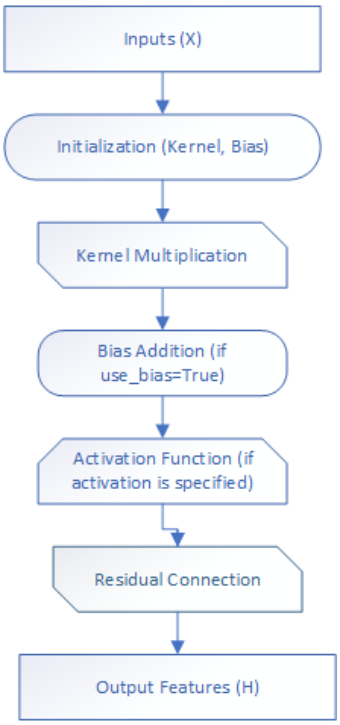


Figure 5.18: Improved GCNLayer

$\underline{NoEmbds}$: This model is identical to the corresponding one in Group 7. It has two improved at the beginning of the model, before the Conv1D architecture and after the embeddings, with parameters: $units = embedding\_dim, activation = 'relu', kernel\_regularizer = tf.keras.regularizers.l2(l2\_reg)$.

$\underline{HoIE}$: This model is identical to the corresponding model in Group 7. It has two improved $GCNLayers$ at the start of the model, like in the $NoEmbds$ model in this Group, with parameters $units = embedding\_dim, activation = 'relu', name = 'gcn\_layer1', kernel\_regularizer = 'l2', use\_bias = True$.

$\underline{DistMult}$: We have again, for this model, two improved $GCNLayers$ at the start of the model with the same parameters as in the $HoIE$ model in this Group. We pass the subject and object embeddings through these two $GCNLayers$, compute the $distmult$ score as in the $DistMult$ model in Group 1, and then we concatenate and feed the result to the Conv1D architecture. The rest of the setup and parameters are the same as in the corresponding model in Group 7.

Group 9

For this Group we use the Conv1D architecture combined with the models in Group 7 and the $GATLayer$.

$\underline{TransE}$: This model is the same as the $TransE$ model in Group 8, except for these: instead of having $GCNLayer$, we use one $GATLayer$ at the start of the model with parameters $num\_heads = 4, output\_dim = 64, activation = 'relu', dropout\_rate = 0.4$.
Additionally, we set the dropout rate to 0.4 for the remaining layers and set the batch size parameter to 128.

$\underline{TransH}$: Here, we introduce an improved version of the $GATLayer$. The initial layer initializes each attention head with the specified output dimension ($output\_dim$) and concatenates all heads' results. This means each attention head operates on the full output dimension, resulting in the final output having a dimension of $num\_heads * output\_dim$. This implementation keeps the output dimension of each head independent of the number of heads, thereby preserving the specified output dimension across all heads.

The improved layer, on the other hand, adjusts the output dimension for each attention head by dividing the specified $output\_dim$ by the number of heads (num_heads). This approach ensures that the concatenated output across all heads equals the specified $output\_dim$. This layer scales down each head's output dimension to ensure the combined result has the desired dimension. This adjustment ensures more fine-grained control over the attention mechanism by distributing the specified output dimension across multiple heads.

Both layers follow similar steps in their call method, where they compute the outputs for each head, concatenate these outputs, apply an activation function, and then a dropout. However, the critical difference lies in how they handle the output dimension for each attention head, influencing the overall architecture and the distribution of computation across the heads.

At the start of this model, we use one improved $GATLayer$ with parameters $num\_heads = 8, output\_dim = 1024, and\ dropout\_rate = 0.5$. The rest of the parameters and setup are the same as in the $TransH$ model in Group 8.

$\underline{RotatE}$: This model is the same as the corresponding model in Group 7, adding to it one improved $GATLayer$ after the second dropout layer and before the flatten layer in Conv1D architecture, with parameters $num\_heads = 8, output\_dim = 1024, activation = 'relu', dropout\_rate = 0.3$.

$\underline{NoEmbds}$: This model is the same as the corresponding model in Group 7, adding to it one improved $GATLayer$ as in the $RotatE$ model in this Group, with parameters $num\_heads = 4, gat\_output\_dim = 256, activation = 'relu', dropout\_rate = 0.3$.

$\underline{HoIE}$: This model is the same as the corresponding model in Group 7, adding to it one $GATLayer$ before the Conv1D architecture, with parameters $num\_heads = 8, output\_dim = 128$.

$\underline{DistMult}$: This model uses subject, predicate, and object embeddings. We concatenate the subject and object embeddings and pass them through an improved $GATLayer$, with parameters $num\_heads = 8, output\_dim = embedding\_dim, activation = 'relu', dropout\_rate = 0.3$. The score is then computed as the reduced sum of the product of the $GATLayer's$ output and the predicate embeddings. Then follows the Conv1D architecture.

Group 10

For this Group we use the Conv1D architecture combined with the models in Group 7, the $GATLayer$ and the $GCNLayer$.

$\underline{TransE}$: We utilize the parameters and setup from the $TransE$ model in Group 9, with the exception of setting the dropout to 0.5. At the start of the model, we use one $GATLayer$ with parameters $num\_heads = gat\_num\_heads, output\_dim = gat\_output\_dim, activation = 'relu', dropout\_rate = dropout\_rate$, and two improved $GCNLayers$ with parameters $units = 256, activation = 'relu'$.

$\underline{RotatE}$: This model is the same as the corresponding model in Group 7, plus one improved $GCNLayer$ and one improved $GATLayer$. The parameters for these layers are $units = 512, activation = 'relu', use\_bias = True,$ and $num\_heads = 8, output\_dim = embedding\_dim * 2, activation = 'relu', dropout\_rate = 0.5$, respectively.

$\underline{NoEmbds}$: This model is the same as the corresponding model in Group 7, plus one improved $GCNLayer$ and one improved $GATLayer$. The parameters for these layers are $units = 256, activation = 'relu', kernel\_regularizer = 'l2'$ and $num\_heads = 4, output\_dim = 256, activation = 'relu', dropout\_rate = 0.3$, respectively.

$\underline{HoIE}$: This model is the same as the corresponding model in Group 7, adding to it one improved $GCNLayer$ and one $GATLayer$ after the second dropout layer and before the flatten of the Conv1D architecture. The parameters for these layers are $units = 400, activation = 'relu'$ and $num\_heads = 4, output\_dim = 64, activation = 'relu', dropout\_rate = 0.5$.

$\underline{DistMult}$: This model is identical to the corresponding model in Group 7, adding one improved $GCNLayer$ and one $GATLayer$ just before the Conv1D architecture. The parameters for these layers are $units = embedding\_dim$ and $num\_heads = 4, output\_dim = embedding\_dim$.

Next, we have the individual experiments. We conducted these experiments by selecting models that yielded positive results and experimenting with various methods or components to observe their impact on the models.

1.4.2 Individual Experiments

Experiment 1

For this experiment, we use Method 2 for negative sample generation, with $corruption\_strength$ set to high, combined with the $DistMult$ model from Group 4, without altering the rest of the model at all.

Experiment 2

For this experiment, we use Method 3 as a method for negative sample generation, combined with the $DistMult$ model from Group 4, without altering the rest of the model at all.

Experiment 3

For this experiment, we use Method 2 for negative sample generation, with $corruption\_strength$ set to high, combined with the $TransH$ model from Group 4, without altering the rest of the model at all.

Experiment 4

For this experiment, we use Method 2 for negative sample generation, with $corruption\_strength$ set to high, combined with the $TransH$ model from Group 6, without altering the rest of the model at all.

Experiment 5

For this experiment, we use Method 3 for negative sample generation, combined with the $TransH$ model from Group 6, without altering the rest of the model at all.

Experiment 6

For this experiment, we use Method 2 for negative sample generation, with $corruption\_strength$ set to high, combined with the $NoEmbds$ model from Group 7, without altering the rest of the model at all.

Experiment 7

For this experiment, we use Method 3 for negative sample generation, combined with the $NoEmbds$ model from Group 7, without altering the rest of the model at all.

Experiment 8

For this experiment, we use Method 3 for negative sample generation, combined with the $HoIE$ model from Group 10, without altering the rest of the model at all.

Experiment 9

For this experiment, we use the $NoEmbds$ model from Group 4 combined with a data augmentation method called flip entities. We designed the $flip\_entities$ function to transform a set of triples by swapping the subject and object entities in each triple. The function begins by initializing an empty list called $flipped\_triples$, which will store the transformed triples. It then iterates over the input list of triples. For each triple, the function constructs a new triple by flipping the subject and object entities, resulting in a triple of the form (o, p, s). The function then appends this new triple to the $flipped\_triples$ list. After processing all triples, the function transforms the list of flipped triples into a NumPy array and then returns it.

In this model, we derive the negative samples using Method 1, and then we pass all the triples to the $flip\_entities$ function. Then, we pass all of the triples to the $flipped\_triples$ function. To assign labels to the augmented triples, we create a set of the initial positive triples for fast membership testing. We check each triple in the augmented dataset against this set, assigning a label of 1 for positive examples and 0 for negative ones.

Experiment 10

For this experiment, we use the model $HoIE$ from Group 8, adding and using the $flip\_entities$ function, just like in Experiment 9.

Experiment 11

For this experiment, we utilize the model from experiment 10, but we use Method 2 for generating negative samples, and we set the $corruption\_strength$ to high.

Experiment 12

For this experiment, we are using the model from experiment 10, but we are using Method 3 for the generation of negative samples.

Experiment 13

In this experiment, we utilize the model from experiment 10 and add a custom attention layer, as shown in Figure 5.19, before the Conv1D architecture begins, using the parameters $units = embedding\_dim$. This layer allows the model to focus on specific parts of the input sequence when making predictions, mimicking the human ability to pay attention to relevant information while disregarding irrelevant details.

Upon initialization, the Attention layer defines three dense (fully connected) sub-layers: W_query, W_key, and V. Each of these sub-layers has a specified number of units, which determines the dimensionality of the transformation applied to the input data. Upon invoking the call method, the layer initially processes the input query through the W_query sub-layer. We use this transformation to extract relevant features from the query to compute attention scores.

We then expand the query vector's dimensions to ensure compatibility with the key vectors, enabling element-wise operations. This step involves reshaping the query vector to facilitate broadcasting across the sequence of values. The attention mechanism computes a score that represents the relevance of each value in the sequence concerning the query. We achieve this by combining the transformed query with the transformed keys, which we obtain by passing the values through W_key and then applying a non-linear activation function (tanh). The V sub-layer generates the final scores.

The $softmax$ function normalizes the computed scores, yielding attention weights. These weights indicate the importance of each value in the sequence relative to the query. We use the attention weights to scale the input values. We multiply each value in the sequence by its corresponding attention weight, then sum the resulting vectors to produce a context vector. This context vector is a weighted sum of the input values, where more relevant values (as determined by the attention weights) contribute more significantly to the context.

The layer then returns the context vector along with the attention weights. The context vector provides a condensed representation of the input sequence, highlighting the most relevant information as dictated by the query. On the other hand, the attention weights provide insight into the parts of the input that the computation deems most important.

Experiment 14

For this experiment, we use the $TransE$ model from Group 10, adding dynamic embeddings to it. Dynamic embeddings are a core innovation in this model. We compute them by taking into account the interaction between the subject (s_embed) and object (o_embed) embeddings. The calculation of dynamic embeddings entails several steps. Firstly, we compute the outer product of the subject and object embeddings, which captures pairwise interactions between their dimensions. An attention layer processes the flattened outer product, allowing the model to focus on the most relevant interactions. We then reshape the attention output back to matrix form and apply a sum operation along the second dimension to aggregate the interactions. Finally, we use a dense layer ($dynamic\_dense$) to non-linearly transform the aggregated interactions, resulting in the final dynamic embeddings.

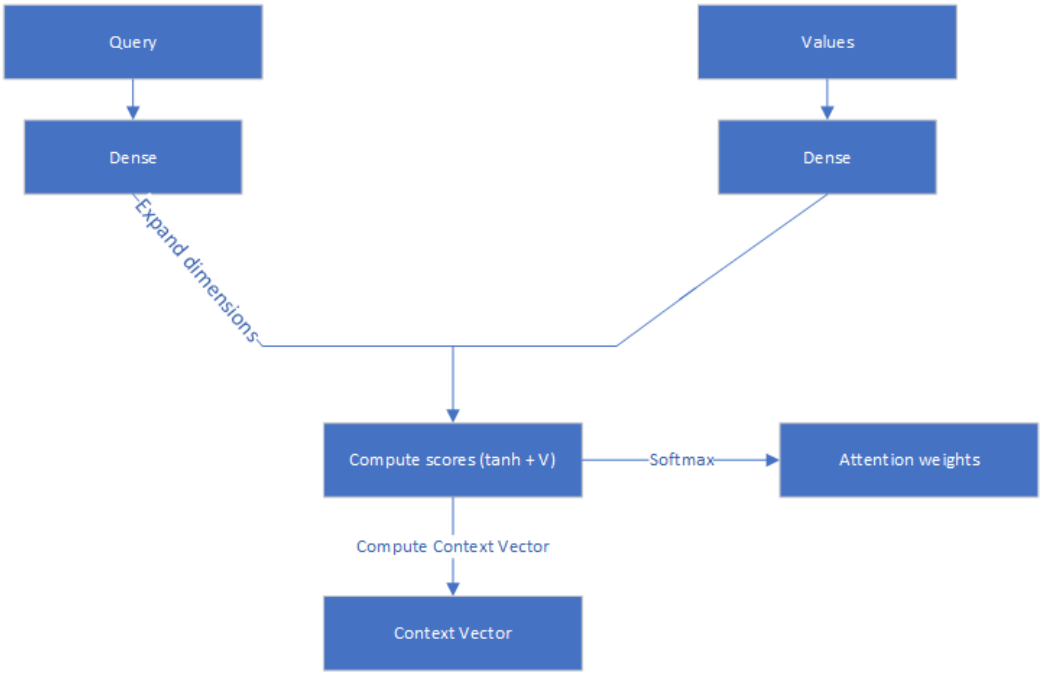


Figure 5.19: Attention Layer for experiment 13

Chapter 6

Results of the experiments

This chapter presents most of the results of all the models from groups and individual experiments. In the current chapter, we present the metrics for each model at the outset, considering three different thresholds (0.5, 0.6, and 0.7), followed by the number of epochs and time required for the model to train. In terms of training time, all experiments were performed on CPUs carried out on a desktop PC with a CPU Core i7-6800k and 32 GB of RAM. The rest of the results are located in Appendix A. The appendix presents the results in the following format: For each experiment group, we start with the metrics for the clustering evaluation. Next, we present the evaluation plots for each model, which include training and validation, precision-recall, probability calibration, confusion matrix, and interpretation plots and tables. Next, we present the statistical data pertaining to the relationships each model predicts. These statistics include both quantitative and qualitative data. Table 6.1 illustrates the quantitative data. The qualitative information includes the top five relations and their frequencies; the relations with a single appearance; and the relations with the highest and lowest average probability.

In Appendix A, the individual experiments start with a clustering evaluation. This is followed by evaluation plots that look like the ones in the groups, then quantitative statistical data about the relationships each model predicts, and finally qualitative statistical data about the relationships each model predicts. For some models, the primary reason for the absence of certain results is their poor performance in saving space within the thesis. For models that performed poorly, we typically include only the training and validation losses, along with a probability calibration plot to illustrate the range of predicted probabilities. However, Chapter 5's GitHub repository contains all the results and code.

Table 6.1: Abbreviations for columns

Column abbreviation	*Column meaning*
C1	Total number of distinct relations
C2	Number of triples with probability > 60%
C3	Number of triples with probability > 90%
C4	Number of distinct relations with probability > 90%
C5	Average probability of all triples
C6	Maximum probability
C7	Minimum probability
C8	Standard deviation of probabilities

Group 1

For this set of experiments, the dense architecture shown in Figure 5.6 is used, along with a simple splitting method and the $TransE, TransH, RotatE, HoIE, DistMult,$ and $ComplEx$ embeddings.

Table 6.2: Metrics for threshold 0.5

	Precision	Recall	F1	ROC AUC	PR -AUC	MCC
$TransE$	0.6307	0.7191	0.6720	0.7639	0.7977	0.3010
$TransH$	0.6342	0.8619	0.7307	0.8100	0.8365	0.3910
$RotatE$	0.6459	0.7071	0.6751	0.7837	0.8134	0.3210
$HoIE$	0.6562	0.6013	0.6276	0.7549	0.7933	0.2874
$DistMult$	0.5000	1.0000	0.6666	0.5000	0.5000	0.0000
$ComplEx$	**0.7009**	**0.8418**	**0.7649**	**0.8224**	**0.7732**	**0.4926**

Table 6.3: Metrics for threshold 0.6

	Precision	Recall	F1	ROC -AUC	PR -AUC	MCC
$TransE$	0.7263	0.5358	0.6166	0.7639	0.7977	0.3460
$TransH$	0.6579	0.7889	0.7175	0.8100	0.8365	0.3865
$RotatE$	0.7643	0.5190	0.6182	0.7837	0.8134	0.3790
$HoIE$	0.7034	0.5384	0.6099	0.7549	0.7933	0.3204
$DistMult$	0.0000	0.0000	0.0000	0.5000	0.5000	0.0000
$ComplEx$	**0.7344**	**0.7337**	**0.7340**	**0.8224**	**0.7732**	**0.4683**

Table 6.4: Metrics for threshold 0.7

	Precision	Recall	F1	ROC -AUC	PR - AUC	MCC
$TransE$	0.8279	0.4493	0.5825	0.7639	0.7977	0.4003
$TransH$	**0.7020**	**0.6908**	**0.6964**	**0.8100**	**0.8365**	**0.3977**
$RotatE$	0.8528	0.4707	0.6066	0.7837	0.8134	0.4357
$HoIE$	0.7659	0.4836	0.5929	0.7549	0.7933	0.3613
$DistMult$	0.0000	0.0000	0.0000	0.5000	0.5000	0.0000
$ComplEx$	**0.7924**	**0.5578**	**0.6547**	**0.8224**	**0.7732**	**0.4311**

Table 6.5: Training times and epochs required for each model

	Seconds per epoch	Total epochs
$TransE$	1300	12
$TransH$	2300	11
$RotatE$	2300	12
$HoIE$	700	11
$DistMult$	1200	16
$ComplEx$	1150	20

Group 2

For this set of experiments, the dense architecture shown in Figure 5.6 is used, along with the stratified K-fold splitting method, $TransE$, $TransH$, $RotatE$, $HoIE$, $DistMult$, and $ComplEx$ embeddings.

Table 6.6: Metrics for threshold 0.5

	Precision	Recall	F1	ROC AUC	PR -AUC	MCC
$TransE$	0.6184	0.7405	0.6740	0.7496	0.7881	0.2892
$TransH$	**0.6541**	**0.8122**	**0.7247**	**0.8121**	**0.8382**	**0.3945**
$RotatE$	0.6421	0.6400	0.6411	0.7597	0.7974	0.2833
$HoIE$	0.6467	0.6300	0.6383	0.7500	0.7898	0.2860
$DistMult$	0.5000	1.0000	0.6667	0.5000	0.5000	0
$ComplEx$	0.6443	0.6352	0.6397	0.7596	0.7969	0.2845

Table 6.7: Metrics for threshold 0.6

	Precision	Recall	F1	ROC -AUC	PR -AUC	MCC
$TransE$	0.6365	0.6819	0.6584	0.7496	0.7881	0.2933
$TransH$	**0.6724**	**0.7707**	**0.7182**	**0.8121**	**0.8382**	**0.3995**
$RotatE$	0.6678	0.5909	0.6270	0.7597	0.7974	0.2990
$HoIE$	0.6737	0.5764	0.6213	0.7500	0.7898	0.3004
$DistMult$	0	0	0	0.5000	0.5000	0
$ComplEx$	0.6681	0.5851	0.6238	0.7596	0.7969	0.2967

Table 6.8: Metrics for threshold 0.7

	Precision	Recall	F1	ROC -AUC	PR - AUC	MCC
$TransE$	0.6614	0.6207	0.6404	0.7496	0.7881	0.3036
$TransH$	**0.6944**	**0.7209**	**0.7074**	**0.8121**	**0.8382**	**0.4039**
$RotatE$	0.7030	0.5430	0.6127	0.7597	0.7974	0.3221
$HoIE$	0.7125	0.5241	0.6039	0.7500	0.7898	0.3241
$DistMult$	0	0	0	0.5000	0.5000	0
$ComplEx$	0.7015	0.5367	0.6081	0.7596	0.7969	0.3172

Table 6.9: Training times and epochs required for each model

	Seconds per epoch	Total epochs	Total folds
$TransE$	450	16	2
$TransH$	1300	12	3
$RotatE$	1550	12	3
$HoIE$	1100	12	3
$DistMult$	750	20	2
$ComplEx$	1400	14	3

Group 3

For this set of experiments, the dense architecture shown in Figure 5.6 is used, along with the Bernouli method for negative sampling generation, a simple splitting method, and the $TransE$, $TransH$, $RotatE$, $HoIE$, $DistMult$, and $ComplEx$ embeddings.

Table 6.10: Metrics for threshold 0.5

	Precision	Recall	F1	ROC- AUC	PR -AUC	MCC
$TransE$	0.5000	1.0000	0.6666	0.5000	0.5000	0
$TransH$	0.5317	0.8767	0.6619	0.6017	0.5881	0.1376
$RotatE$	**0.5303**	**0.8460**	**0.6520**	**0.5962**	**0.5913**	**0.1206**
$HoIE$	0.5394	0.7512	0.6279	0.6056	0.5932	0.1196
$DistMult$	0.5363	0.8043	0.6435	0.6029	0.5922	0.1259
$ComplEx$	0.5000	1.0000	0.6666	0.5000	0.5000	0

Table 6.11: Metrics for threshold 0.6

	Precision	Recall	F1	ROC-AUC	PR -AUC	MCC

$TransE$	0	0	0	0.5000	0.5000	0
$TransH$	0.5501	0.5917	0.5701	0.6017	0.5881	0.1081
$RotatE$	0.6218	0.3493	0.4473	0.5962	0.5913	0.1523
$HoIE$	0.6542	0.2832	0.3952	0.6056	0.5932	0.1621
$DistMult$	**0.6009**	**0.3900**	**0.4730**	**0.6029**	**0.5922**	**0.1399**
$ComplEx$	0	0	0	0.5000	0.5000	0

Table 6.12: Metrics for threshold 0.7

	Precision	**Recall**	**F1**	**ROC-AUC**	**PR - AUC**	**MCC**
$TransE$	0	0	0	0.5000	0.5000	0
$TransH$	0.6459	0.0209	0.0405	0.6017	0.5881	0.0375
$RotatE$	**0.6713**	**0.1572**	**0.2547**	**0.5962**	**0.5913**	**0.1248**
$HoIE$	0.6745	0.0785	0.1406	0.6056	0.5932	0.0867
$DistMult$	0.6797	0.0677	0.1232	0.6029	0.5922	0.0823
$ComplEx$	0	0	0	0.5000	0.5000	0

Table 6.13: Training times and epochs required for each model

	Seconds per epoch	**Total epochs**
$TransE$	800	15
$TransH$	800	12
$RotatE$	1400	12
$HoIE$	750	12
$DistMult$	900	12
$ComplEx$	1100	14

Group 4

For this set of experiments, the dense architecture shown in Figure 5.6 is used, along with Method 1 for negative sampling generation, the StratifiedShuffleSplit splitting method, the $GCNLayer$, and the $TransE$, $TransH$, $NoEmbds$, $HoIE$, $DistMult$, and $ComplEx$ embeddings.

Table 6.14: Metrics for threshold 0.5

	Precision	**Recall**	**F1**	**ROC- AUC**	**PR -AUC**	**MCC**

$TransE$	0.6505	0.7725	0.7063	0.7993	0.8255	0.3639
$TransH$	0.6421	0.8259	0.7225	0.7974	0.8241	0.3815
$NoEmbds$	0.7668	0.5506	0.6409	0.8062	0.8305	0.3994
$HoIE$	**0.6454**	**0.8760**	**0.7432**	**0.8103**	**0.8335**	**0.4225**
$DistMult$	**0.7459**	**0.6286**	**0.6822**	**0.8247**	**0.8450**	**0.4196**
$ComplEx$	0.6766	0.7791	0.7243	0.8225	0.8442	0.4115

Table 6.15: Metrics for threshold 0.6

	Precision	**Recall**	**F1**	**ROC-AUC**	**PR -AUC**	**MCC**
$TransE$	0.8428	0.4894	0.6192	0.7993	0.8255	0.4385
$TransH$	0.8378	0.4933	0.6210	0.7974	0.8241	0.4364
$NoEmbds$	0.8570	0.4858	0.6201	0.8062	0.8305	0.4491
$HoIE$	**0.8685**	**0.4869**	**0.6240**	**0.8103**	**0.8335**	**0.4600**
$DistMult$	**0.9015**	**0.4693**	**0.6172**	**0.8247**	**0.8450**	**0.4763**
$ComplEx$	0.8948	0.4866	0.6304	0.8225	0.8442	0.4825

Table 6.16: Metrics for threshold 0.7

	Precision	**Recall**	**F1**	**ROC-AUC**	**PR - AUC**	**MCC**
$TransE$	0.8873	0.4720	0.6162	0.7993	0.8255	0.4663
$TransH$	0.8732	0.4760	0.6161	0.7974	0.8241	0.4569
$NoEmbds$	0.8918	0.4714	0.6168	0.8062	0.8305	0.4697
$HoIE$	0.8989	0.4716	0.6186	0.8103	0.8335	0.4757
$DistMult$	**0.9663**	**0.4271**	**0.5924**	**0.8247**	**0.8450**	**0.4968**
$ComplEx$	**0.9244**	**0.4737**	**0.6264**	**0.8225**	**0.8442**	**0.4982**

Table 6.17: Training times and epochs required for each model

	Seconds per epoch	**Total epochs**
$TransE$	1150	5
$TransH$	760	5
$NoEmbds$	800	5
$HoIE$	1300	5
$DistMult$	1300	5
$ComplEx$	1100	5

For this set of experiments, the dense architecture shown in Figure 5.6 is used, along with Method 1 for negative sampling generation, the StratifiedShuffleSplit splitting method, the $GATLayer$, and the $TransE$, $TransH$, $RatatE$, $HoIE$, $DistMult$, and $ComplEx$ embeddings.

Table 6.18: Metrics for threshold 0.5

	Precision	Recall	F1	ROC- AUC	PR -AUC	MCC
$TransE$	0.6643	0.8601	0.7496	0.8265	0.8456	0.4452
$TransH$	**0.6672**	**0.8937**	**0.7640**	**0.8373**	**0.8550**	**0.4763**
$RotatE$	0.5000	1.0000	0.6666	0.5000	0.5000	0
$HoIE$	0.6397	0.9433	0.7624	0.8290	0.8471	0.4682
$DistMult$	1.0000	0.0221	0.0433	0.5139	0.5329	0.1058
$ComplEx$	0.6552	0.9017	0.7590	0.8314	0.8500	0.4612

Table 6.19: Metrics for threshold 0.6

	Precision	Recall	F1	ROC-AUC	PR -AUC	MCC
$TransE$	0.8539	0.5100	0.6386	0.8265	0.8456	0.4619
$TransH$	**0.9315**	**0.4846**	**0.6375**	**0.8373**	**0.8550**	**0.5117**
$RotatE$	0	0	0	0.5000	0.5000	0
$HoIE$	0.8915	0.5013	0.6417	0.8290	0.8471	0.4897
$DistMult$	1.0000	0.0221	0.0433	0.5139	0.5329	0.1058
$ComplEx$	0.9388	0.4692	0.6257	0.8314	0.8500	0.5066

Table 6.20: Metrics for threshold 0.7

	Precision	Recall	F1	ROC-AUC	PR - AUC	MCC
$TransE$	0.9811	0.3999	0.5682	0.8265	0.8456	0.4868
$TransH$	0.9922	0.3857	0.5554	0.8373	0.8550	0.4835
$RotatE$	0	0	0	0.5000	0.5000	0
$HoIE$	**0.9614**	**0.4374**	**0.6013**	**0.8290**	**0.8471**	**0.5008**
$DistMult$	1.0000	0.0221	0.0433	0.5139	0.5329	0.1058
$ComplEx$	0.9944	0.3217	0.4861	0.8314	0.8500	0.4343

Table 6.21: Training times and epochs required for each model

	Seconds per epoch	Total epochs

$TransE$	1450	8
$TransH$	1450	7
$RotatE$	2450	7
$HoIE$	4000	7
$DistMult$	1450	7
$ComplEx$	3300	5

Group 6

For this set of experiments, the dense architecture shown in Figure 5.6 is used, along with Method 1 for negative sampling generation, the StratifiedShuffleSplit splitting method, the $GCNLayer$, the $GATLayer$, and the $TransE$, $TransH$, $RatatE$, $HoIE$, $DistMult$, and $ComplEx$ embeddings.

Table 6.22: Metrics for threshold 0.5

	Precision	Recall	F1	ROC- AUC	PR -AUC	MCC
$TransE$	**0.6500**	**0.9171**	**0.7608**	**0.8279**	**0.8469**	**0.4643**
$TransH$	0.6580	0.8845	0.7547	0.8293	0.8481	0.4526
$RotatE$	0.5264	0.9182	0.6692	0.7060	0.7722	0.1383
$HoIE$	**0.7031**	**0.7637**	**0.7321**	**0.8351**	**0.8527**	**0.4429**
$DistMult$	0.5000	1.0000	0.6666	0.7583	0.7809	0
$ComplEx$	0.5000	1.0000	0.6666	0.6682	0.7560	0

Table 6.23: Metrics for threshold 0.6

	Precision	Recall	F1	ROC-AUC	PR -AUC	MCC
$TransE$	**0.9256**	**0.4690**	**0.6226**	**0.8279**	**0.8469**	**0.4958**
$TransH$	0.9801	0.3975	0.5656	0.8293	0.8481	0.4843
$RotatE$	0.8846	0.4129	0.5630	0.7060	0.7722	0.4245
$HoIE$	**0.9507**	**0.4517**	**0.6124**	**0.8351**	**0.8527**	**0.5031**
$DistMult$	0.5475	0.9966	0.7068	0.7583	0.7809	0.3026
$ComplEx$	0.6005	0.5832	0.5917	0.6682	0.7560	0.1954

Table 6.24: Metrics for threshold 0.7

	Precision	Recall	F1	ROC-AUC	PR - AUC	MCC
$TransE$	**0.9948**	**0.3453**	**0.5127**	**0.8279**	**0.8469**	**0.4535**
$TransH$	0.9985	0.2722	0.4278	0.8293	0.8481	0.3961
$RotatE$	0.9888	0.3397	0.5057	0.7060	0.7722	0.4453
$HoIE$	**0.9976**	**0.3110**	**0.4742**	**0.8351**	**0.8527**	**0.4277**
$DistMult$	0.6165	0.8771	0.7240	0.7583	0.7809	0.3658
$ComplEx$	0.9367	0.4184	0.5784	0.6682	0.7560	0.4684

Table 6.25: Training times and epochs required for each model

	Seconds per epoch	Total epochs
$TransE$	1370	7
$TransH$	1350	7
$RotatE$	2500	6
$HoIE$	1550	7
$DistMult$	1670	7
$ComplEx$	3300	5

Group 7

For this set of experiments, the Conv1D architecture shown in Figure 5.7 is used, along with Method 1 for negative sampling generation, the StratifiedShuffleSplit splitting method, and the $TransE$, $TransH$, $RatatE$, $NoEmbds$, $HoIE$, $DistMult$, and $ComplEx$ embeddings.

Table 6.26: Metrics for threshold 0.5

	Precision	Recall	F1	ROC- AUC	PR - AUC	MCC
$TransE$	0.7416	0.5879	0.6559	0.8096	0.8336	0.3915
$TransH$	0.5000	1.0000	0.6667	0.5000	0.5000	0
$RotatE$	0.5046	0.4224	0.4598	0.5089	0.5260	0.0078
$NoEmbds$	**0.7022**	**0.7329**	**0.7172**	**0.8287**	**0.8490**	-
$HoIE$	0.7153	0.5876	0.6452	0.8035	0.8297	0.3595
$DistMult$	0.5433	0.3597	0.4329	0.5391	0.5685	0.0610
$ComplEx$	0.5000	1.0000	0.6667	0.5000	0.5000	0

Table 6.27: Metrics for threshold 0.6

	Precision	Recall	F1	ROC-AUC	PR - AUC	MCC
$TransE$	0.8770	0.4993	0.6363	0.8096	0.8336	0.4756
$TransH$	0	0	0	0.5000	0.5000	0
$RotatE$	0.9990	0.0223	0.0436	0.5089	0.5260	0.1060
$NoEmbds$	**0.7655**	**0.6056**	**0.6762**	**0.8287**	**0.8490**	-
$HoIE$	0.8623	0.4969	0.6305	0.8035	0.8297	0.4610
$DistMult$	0.5480	0.3338	0.4148	0.5391	0.5685	0.0635
$ComplEx$	0	0	0	0.5000	0.5000	0

Table 6.28: Metrics for threshold 0.7

	Precision	Recall	F1	ROC-AUC	PR - AUC	MCC
$TransE$	0.9329	0.4696	0.6247	0.8096	0.8336	0.5021
$TransH$	0	0	0	0.5000	0.5000	0
$RotatE$	1.0000	0.0222	0.0435	0.5089	0.5260	0.1061
$NoEmbds$	**0.8841**	**0.5024**	**0.6407**	**0.8287**	**0.8490**	-
$HoIE$	0.9312	0.4740	0.6282	0.8035	0.8297	0.5038
$DistMult$	0.5546	0.3069	0.3951	0.5391	0.5685	0.0675
$ComplEx$	0	0	0	0.5000	0.5000	0

Table 6.29: Training times and epochs required for each model

	Seconds per epoch	Total epochs
$TransE$	8800	7
$TransH$	8300	13
$RotatE$	2850	5
$NoEmbds$	8500	6
$HoIE$	11500	6
$DistMult$	860	5
$ComplEx$	8900	11

Group 8

For this set of experiments, the Conv1D architecture shown in Figure 5.7 is used, along with Method 1 for negative sampling generation, the StratifiedShuffleSplit splitting method, the $GCNLayer$, and the $TransE$, $TransH$, $RatatE$, $NoEmbds$, $HoIE$, $DistMult$, and $ComplEx$ embeddings.

Table 6.30: Metrics for threshold 0.5

	Precision	Recall	F1	ROC- AUC	PR -AUC	MCC
$TransE$	0.6821	0.6085	0.6432	0.7919	0.8182	-
$TransH$	0	0	0	0.5	0.5	0
$RotatE$	0.5110	0.6422	0.5691	0.5177	0.5116	0.0286
$NoEmbds$	**0.8300**	**0.5363**	**0.6516**	**0.8282**	**0.8487**	-
$HoIE$	0.6754	0.6527	0.6639	0.7576	0.7491	0.3392
$DistMult$	0.6675	0.5880	0.6252	0.7003	0.7523	0.2972
$ComplEx$	0.5000	1.0000	0.6667	0.5000	0.5000	0

Table 6.31: Metrics for threshold 0.6

	Precision	Recall	F1	ROC-AUC	PR -AUC	MCC
$TransE$	0.9081	0.4596	0.6103	0.7919	0.8182	-
$TransH$	0	0	0	0.5	0.5	0
$RotatE$	0	0	0	0.5177	0.5116	0
$NoEmbds$	**0.8971**	**0.4941**	**0.6372**	**0.8282**	**0.8487**	-
$HoIE$	0.6923	0.6045	0.6454	0.7576	0.7491	0.3386
$DistMult$	0.6794	0.5729	0.6216	0.7003	0.7523	0.3063
$ComplEx$	0	0	0	0.5	0.5	0

Table 6.32: Metrics for threshold 0.7

	Precision	Recall	F1	ROC-AUC	PR - AUC	MCC
$TransE$	0.9550	0.4349	0.5976	0.7919	0.8182	-
$TransH$	0	0	0	0.5	0.5	0
$RotatE$	0	0	0	0.5177	0.5116	0
$NoEmbds$	**0.9326**	**0.4768**	**0.6310**	**0.8282**	**0.8487**	-
$HoIE$	0.7113	0.5642	0.6293	0.7576	0.7491	0.3427
$DistMult$	0.6927	0.5577	0.6179	0.7003	0.7523	0.3164
$ComplEx$	0	0	0	0.5	0.5	0

Table 6.33: Training times and epochs required for each model

	Seconds per epoch	Total epochs
$TransE$	16000	5
$TransH$	9000	15
$RotatE$	1800	5
$NoEmbds$	8300	5
$HoIE$	15000	7
$DistMult$	1550	5

Group 9

For this set of experiments, the Conv1D architecture shown in Figure 5.7 is used, along with Method 1 for negative sampling generation, the StratifiedShuffleSplit splitting method, the $GATLayer$, and the $TransE$, $TransH$, $RatatE$, $NoEmbds$, $HoIE$, and $DistMult$ embeddings.

Table 6.34: Metrics for threshold 0.5

	Precision	Recall	F1	ROC- AUC	PR -AUC	MCC
$TransE$	0.5017	1.0000	0.6682	0.7031	0.6850	-
$TransH$	0.5000	1.0000	0.6667	0.5000	0.5000	0
$RotatE$	0.5218	0.2706	0.3564	0.5156	0.5342	0.0258
$NoEmbds$	0.7577	0.5316	0.6249	0.7943	0.8226	0.3789
$HoIE$	**0.8870**	**0.4647**	**0.6099**	**0.8159**	**0.8358**	**0.4611**
$DistMult$	0.6902	0.6535	0.6713	0.7873	0.8216	0.3607

Table 6.35: Metrics for threshold 0.6

	Precision	Recall	F1	ROC-AUC	PR -AUC	MCC
$TransE$	0.5022	1.0000	0.6686	0.7031	0.6850	-
$TransH$	0	0	0	0.5000	0.5000	0
$RotatE$	0.8947	0.0249	0.0484	0.5156	0.5342	0.0937
$NoEmbds$	0.9108	0.4674	0.6178	0.7943	0.8226	0.4827
$HoIE$	**0.9582**	**0.4160**	**0.5801**	**0.8159**	**0.8358**	**0.4825**
$DistMult$	0.7017	0.6359	0.6672	0.7873	0.8216	0.3672

Table 6.36: Metrics for threshold 0.7

	Precision	Recall	F1	ROC-AUC	PR - AUC	MCC
$TransE$	0.5028	1.0000	0.6691	0.7031	0.6850	-
$TransH$	0	0	0	0.5000	0.5000	0
$RotatE$	1.0000	0.0214	0.0420	0.5156	0.5342	0.1041
$NoEmbds$	0.9542	0.4474	0.6092	0.7943	0.8226	0.5027
$HoIE$	**0.9753**	**0.3844**	**0.5514**	**0.8159**	**0.8358**	**0.4709**
$DistMult$	0.7158	0.6175	0.6630	0.7873	0.8216	0.3758

Table 6.37: Training times and epochs required for each model

	Seconds per epoch	Total epochs
$TransE$	20500	5
$TransH$	5500	15
$RotatE$	1900	5
$NoEmbds$	11000	5
$HoIE$	6600	6
$DistMult$	950	7

Group 10

For this set of experiments, the Conv1D architecture shown in Figure 5.7 is used, along with Method 1 for negative sampling generation, the StratifiedShuffleSplit splitting method, the $GCNLayer$, the $GATLayer$, and the $TransE$, $RatatE$, $NoEmbds$, $HoIE$, and $DistMult$ embeddings.

Table 6.38: Metrics for threshold 0.5

	Precision	Recall	F1	ROC- AUC	PR -AUC	MCC
$TransE$	0.6763	0.5814	0.6252	0.7613	0.7923	0.3061
$RotatE$	0.5095	0.8218	0.6291	0.5236	0.5148	0.0390
$NoEmbds$	**0.7898**	**0.5499**	**0.6484**	**0.8168**	**0.8387**	-
$HoIE$	**0.6478**	**0.7789**	**0.7073**	**0.7994**	**0.8276**	**0.3630**
$DistMult$	0.7169	0.5207	0.6033	0.7717	0.7788	0.3276

Table 6.39: Metrics for threshold 0.6

	Precision	Recall	F1	ROC-AUC	PR -AUC	MCC
$TransE$	0.7266	0.4905	0.5857	0.7613	0.7923	0.3235
$RotatE$	0.5126	0.7281	0.6016	0.5236	0.5148	0.0395
$NoEmbds$	**0.9301**	**0.4660**	**0.6209**	**0.8168**	**0.8387**	-
$HoIE$	**0.7623**	**0.5407**	**0.6327**	**0.7998**	**0.8267**	**0.3889**
$DistMult$	0.7431	0.4679	0.5742	0.7717	0.7788	0.3296

Table 6.40: Metrics for threshold 0.7

	Precision	Recall	F1	ROC-AUC	PR - AUC	MCC
$TransE$	0.7828	0.4559	0.5762	0.7613	0.7923	0.3626
$RotatE$	0.5172	0.4266	0.4675	0.5236	0.5148	0.0289
$NoEmbds$	**0.9566**	**0.4504**	**0.6125**	**0.8168**	**0.8387**	-
$HoIE$	**0.9032**	**0.4820**	**0.6285**	**0.7998**	**0.8267**	**0.4865**
$DistMult$	0.7822	0.4075	0.5359	0.7717	0.7788	0.3350

Table 6.41: Training times and epochs required for each model

	Seconds per epoch	Total epochs
$TransE$	25500	7
$RotatE$	1800	7
$NoEmbds$	16500	5
$HoIE$	20500	6
$DistMult$	5900	7

Experiment 1

For this experiment, we use Method 2 for negative sample generation, with $corruption\_strength$ set to high, combined with the $DistMult$ model from Group 4.

Table 6.42: Metrics for different thresholds

	Precision	Recall	F1	ROC - AUC	PR - AUC	MCC
Threshold > 0.5	0.5077	0.1992	0.2862	0.6774	0.3888	0.1963
Threshold > 0.6	0.5470	0.0173	0.0335	0.6774	0.3888	0.0612
Threshold > 0.7	0	0	0	0.6774	0.3888	-0.0013

Table 6.43: Training time and epoch required for the model

Seconds per epoch	Total epochs
1550	5

Experiment 2

For this experiment, we use Method 3 as a method for negative sample generation, combined with the $DistMult$ model from Group 4.

Table 6.44: Metrics for different thresholds

	Precision	Recall	F1	ROC-AUC	PR - AUC	MCC
Threshold > 0.5	0.7865	0.9500	0.8605	0.9667	0.9702	0.7075
Threshold > 0.6	0.8092	0.9297	0.8653	0.9667	0.9702	-
Threshold > 0.7	0.8427	0.9075	0.8739	0.9667	0.9702	-

Table 6.45: Training time and epoch required for the model

Seconds per epoch	Total epochs
850	5

Experiment 3

For this experiment, we use Method 2 for negative sample generation, with $corruption\_strength$ set to high, combined with the $TransH$ model from Group 4.

Table 6.46: Metrics for different thresholds

	Precision	Recall	F1	ROC-AUC	PR - AUC	MCC
Threshold > 0.5	0.5638	0.2256	0.3222	0.7290	0.4412	0.2416
Threshold > 0.6	0	0	0	0.7290	0.4412	0
Threshold > 0.7	0	0	0	0.7290	0.4412	0

Table 6.47: Training time and epoch required for the model

Seconds per epoch	Total epochs
1590	5

Experiment 4

For this experiment, we use Method 2 for negative sample generation, with $corruption\_strength$ set to high, combined with the $TransH$ model from Group 6.

Table 6.48: Metrics for different thresholds

	Precision	Recall	F1	ROC- AUC	PR -AUC	MCC
Threshold > 0.5	0	0	0	0.7220	0.4201	0

Table 6.49: Training time and epoch required for the model

Seconds per epoch	Total epochs
3100	7

Experiment 5

For this experiment, we use Method 3 for negative sample generation, combined with the $TransH$ model from Group 6.

Table 6.50: Metrics for different thresholds

	Precision	Recall	F1	ROC-AUC	PR - AUC	MCC
Threshold > 0.5	0.9930	0.9883	0.9906	0.9998	0.9998	0.9813
Threshold > 0.6	0.9987	0.9715	0.9849	0.9998	0.9998	0.9705
Threshold > 0.7	1.0000	0.9499	0.9743	0.9998	0.9998	0.9510

Table 6.51: Training time and epoch required for the model

Seconds per epoch	Total epochs
1500	13

Experiment 6

For this experiment, we use Method 2 for negative sample generation, with $corruption\_strength$ set to high, combined with the $NoEmbds$ model from Group 7.

Table 6.52: Metrics for different thresholds

	Precision	Recall	F1	ROC-AUC	PR - AUC	MCC
Threshold > 0.5	0.6412	0.3733	0.4719	0.8081	0.5925	-
Threshold > 0.6	0.7219	0.2817	0.4053	0.8081	0.5925	-
Threshold > 0.7	0.7882	0.1962	0.3142	0.8081	0.5925	-

Table 6.53: Training time and epoch required for the model

Seconds per epoch	Total epochs
17400	7

Experiment 7

For this experiment, we use Method 3 for negative sample generation, combined with the $NoEmbds$ model from Group 7.

Table 6.54: Metrics for different thresholds

	Precision	Recall	F1	ROC - AUC	PR - AUC	MCC
Threshold > 0.5	0.9942	0.9761	0.9851	0.9990	0.9990	0.9706
Threshold > 0.6	0.9949	0.9682	0.9814	0.9990	0.9990	0.9637
Threshold > 0.7	0.9956	0.9576	0.9762	0.9990	0.9990	0.9540

Table 6.55: Training time and epoch required for the model

Seconds per epoch	Total epochs
8400	5

Experiment 8

For this experiment, we use Method 3 for negative sample generation, combined with the $HoIE$ model from Group 10.

Table 6.56: Metrics for different thresholds

	Precision	Recall	F1	ROC - AUC	PR - AUC	MCC
Threshold > 0.5	0.9929	0.9794	0.9861	0.9991	0.9991	0.9725
Threshold > 0.6	0.9936	0.9730	0.9832	0.9991	0.9991	0.9669
Threshold > 0.7	0.9943	0.9643	0.9790	0.9991	0.9991	0.9591

Table 6.57: Training time and epoch required for the model

Seconds per epoch	Total epochs
20500	5

Experiment 9

For this experiment, we use the $NoEmbds$ model from Group 4 combined with the data augmentation method called flip entities.

Table 6.58: Metrics for different thresholds

	Precision	Recall	F1	ROC - AUC	PR - AUC	MCC
Threshold > 0.5	0.9970	0.2929	0.4528	0.9264	0.8334	0.4839
Threshold > 0.6	0.9994	0.0341	0.0659	0.9264	0.8334	0.1597
Threshold > 0.7	1.0000	0.0031	0.0062	0.9264	0.8334	0.0482

Table 6.59: Training time and epoch required for the model

Seconds per epoch	Total epochs
3150	5

Experiment 10

For this experiment, we use the model $HoIE$ from Group 8, adding and using the $flip\_entities$ function, just like in Experiment 9.

Table 6.60: Metrics for different thresholds

	Precision	Recall	F1	ROC - AUC	PR - AUC	MCC
Threshold > 0.5	0.7297	0.7866	0.7571	0.9441	0.8682	0.6695
Threshold > 0.6	0.8105	0.6472	0.7197	0.9441	0.8682	0.6435
Threshold > 0.7	0.9292	0.5129	0.6609	0.9441	0.8682	0.6253

Table 6.61: Training time and epoch required for the model

Seconds per epoch	Total epochs
7900	5

Experiment 11

For this experiment, we utilize the model from experiment 10, but we use Method 2 for generating negative samples, and we set the $corruption\_strength$ to high.

Table 6.62: Metrics for different thresholds

	Precision	Recall	F1	ROC - AUC	PR - AUC	MCC
Threshold > 0.5	0.6783	0.4923	0.5705	0.9310	0.6697	0.5268
Threshold > 0.6	0.7540	0.3949	0.5184	0.9310	0.6697	0.5021
Threshold > 0.7	0.8134	0.2891	0.4265	0.9310	0.6697	0.4477

Table 6.63: Training time and epoch required for the model

Seconds per epoch	Total epochs
15000	5

Experiment 12

For this experiment, we are using the model from experiment 10, but we are using Method 3 for the generation of negative samples.

Table 6.64: Metrics for different thresholds

	Precision	Recall	F1	ROC - AUC	PR - AUC	MCC
Threshold > 0.5	0.9863	0.9820	0.9841	0.9979	0.9968	0.9786
Threshold > 0.6	0.9873	0.9806	0.9839	0.9979	0.9968	0.9784
Threshold > 0.7	0.9880	0.9788	0.9834	0.9979	0.9968	0.9776

Table 6.65: Training time and epoch required for the model

Seconds per epoch	Total epochs
7400	13

Experiment 13

In this experiment, we utilize the model from experiment 10 and add a custom attention layer, as shown in Figure 5.19, before the Conv1D architecture begins.

Table 6.66: Metrics for different thresholds

	Precision	Recall	F1	ROC - AUC	PR - AUC	MCC
Threshold > 0.5	0	0	0	0.5000	0.2572	0

Table 6.67: Training time and epoch required for the model

Seconds per epoch	Total epochs
7800	15

Experiment 14

For this experiment, we use the $TransE$ model from Group 10, adding dynamic embeddings to it.

Table 6.68: Metrics for different thresholds

	Precision	Recall	F1	ROC - AUC	PR - AUC	MCC
Threshold > 0.5	0.6466	0.6434	0.6450	0.7597	0.7925	0.2917
Threshold > 0.6	0.7018	0.5327	0.6057	0.7597	0.7925	0.3157
Threshold > 0.7	0.7466	0.4755	0.5810	0.7597	0.7925	0.3372

Table 6.69: Training time and epoch required for the model

Seconds per epoch	Total epochs
30500	7

2.1 Discussion

After analyzing the results, we can draw several intriguing conclusions about the models. We can examine how the various components and methods impact the efficiency and training time of the models. Furthermore, we can evaluate how well the models performed in the relation prediction task using our approach, as outlined in Chapter 5. In every set of experiments, we highlight the top-performing model in bold. In certain experiments, there may be multiple models highlighted in bold. This is because certain models may outperform others in certain metrics, while others excel in the rest of the metrics. In order to determine the best model, we examine its performance across the metrics and compare it to the other models. Beginning with the triple classification task, which serves as the primary objective for all models, it is evident that we have successfully accomplished the task. In fact, we have achieved nearly flawless results in certain experiments.

Upon examining the aggregated results, it becomes evident that there is a wide range of both quantitative and qualitative findings. The initial set of models, whose results are shown in Tables 6.2, 6.3, and 6.4, utilized $ComplEx$-like embeddings and incorporated skip connections, resulting in improved and well-balanced performance across all three thresholds. The $TransH$-like embeddings model also achieved a well-balanced performance across the metrics for the 0.7 threshold. It is worth mentioning that all the models in this group performed admirably, with only minor variations in the metrics. Overall, the model using $DistMult$-like embeddings did not perform well. The graphs clearly demonstrate the

significant impact of adding dropout layers, normalization, and early stopping on preventing overfitting and creating models suitable for a wide range of scenarios.

When examining the accuracy of the probabilities derived from the calibration plots, it becomes apparent that for most models, particularly for lower probabilities (<= 0.5), which are typically less significant, the models tend to lack confidence. However, when it comes to probabilities greater than 0.5, the models tend to be overly confident. The $ComplEx$ models that excelled in this group demonstrated nearly flawless calibration for probabilities ranging from 0.5 to 0.9, but showed a slight tendency to be overly confident for probabilities exceeding 0.9. The precision-recall curves exhibit similar patterns across all models, with the exception of the $ComplEx$ models, which demonstrate a slightly improved curve for higher recall values. By analyzing the interpretation plots, we can determine which component of the triple has the greatest impact on the probability we select. In most models, the relationship has a significant impact on the probability. When analyzing the $TransH$ model, it becomes evident that both the object and subject have a noticeable impact, although it is not as significant as the impact of the relation. In the $ComplEx$ model, the probability is greatly influenced by both the subject and the object, sometimes even more so than the relationship itself.

By examining the impact of each component of the triplet on the five specific triples, we can verify that the impacts are consistent with those depicted in the interpretation graphs. We can now observe that all of the models provide satisfactory metrics in the relation prediction assignment, with the $RotatE$ model performing slightly better. The $TransH$ model produces the most diverse results from the statistics of the recommended triplets, predicting 29 distinct relations, 16 of which have a very high probability exceeding 0.9. In this model, the average probability of triplets is high, exceeding 0.5. The $owl\#imports$ and the $owl\#equivalentClass$ are the two most frequently predicted relations, with frequencies of 852 and 374, respectively. Furthermore, the $TransE$ model contained numerous distinct relationships with a probability greater than 0.9 in comparison to the total number of distinct predicted relations (7 out of 9). It is noteworthy that all of the models for this group produced triples and relations with high probabilities. According to Table 6.5, the average training time and epochs required for the models in this group are 1500 seconds per epoch and 14 epochs, respectively.

For all the different thresholds we used, Group 2's $TransH$-like embeddings model nearly outperformed all other models, as shown in Tables 6.6, 6.7, and 6.8. All models performed similarly across all metrics, except for the $DistMult$ model, which again performed poorly in the aggregate. The models' graphs are quite similar to those in Group 1. The $TransH$ model exhibits a significantly superior precision-recall curve and calibration plot in comparison to the other models. The calibration approaches the ideal calibration for probabilities above 0.9, which are relevant to us, despite the fact that all models for probabilities over 0.5 are overconfident. For the new recommended triples, the models predict relationships with probabilities exceeding 0.9. However, they do not make divergent predictions, predicting a maximum of four distinct relations.

The $ComplEx$ model is an exception, as it outperformed the other models in terms of clustering metrics and predicts 14 distinct relations, six of which have a probability greater

than 0.9. The alternative division strategy does not appear to enhance the models' performance. Folds were used in the dividing procedure, resulting in a significant increase in training time. In this group, according to Table 6.9, the average training time for the models is 1200 seconds per epoch and 14 epochs for each fold, respectively.

In general, the results of Group 3, shown in Tables 6.10, 6.11, and 6.12, are inferior to those of the preceding two categories. The $TransE$ and the $ComplEx$ are two models that demonstrated subpar aggregate performance. The $RotatE$ model, which used the Fourier transform, performed better than the other models. In this group, the results of the $DistMult$ model are in stark contrast to those of the previous categories, which outperformed the clustering metrics. The training and validation diagrams are quite similar to those of the previous groups. The precision-recall curves are significantly worse than those of the previous groups. It is important to note that none of the models predict probabilities exceeding 0.9; however, they exhibit relatively excellent calibration.

The interpretation diagrams' conclusion indicates that the relationship continues to significantly influence the probabilities. However, there are cases where the subject and object also significantly influence the triples' probabilities. The predicted triple statistics show that none of the models predict relationships with probabilities greater than 0.9. The $TransH$ model predicts 576 out of 1500 predicted triplets, with a probability greater than 0.6 and a maximal probability of 0.86. It predicts 8 distinct relations. The $DistMult$ model, which outperformed the other models, predicts 25 distinct relations with 75 triples with a probability greater than 0.6 and a maximal probability of.68. The alternative negative sampling generation strategy does not appear to enhance the models' performance. In this group, according to Table 6.13, the average training time for the models is 1100 seconds per epoch and 13 epochs, respectively.

In Group 4, the results of which are shown in the Tables 6.14, 6.15, and 6.16, where we employ the $GCNLayer$, the $HoIE$ and $DistMult$ models outperform the other models in terms of metrics at thresholds of 0.5 and 0.6. Setting the threshold value to 0.7 results in superior performance across all metrics for the $ComplEx$ and $DistMult$ models. It is evident that the models in this group exhibit superior performance in comparison to those in Groups 2 and 3. When compared to Group 1, the metrics for Group 1 are more evenly distributed, as evidenced by the nearly equal precision and recall. The $ComplEx$ models appear to outperform the other models in terms of the clustering metrics. We can observe that the models have a propensity to overfit when we examine the plots, beginning with the training and validation plots. The early halting halts the training process in a timely manner, but the absence of dropout layers is evident.

All the precision-recall curves, with the exception of those in the $TransH$ and $NoEmbds$ models, are essentially identical. All the models, except for $DistMult$, share a common feature in their calibration plots for probabilities. It's perfectly calibrated for the low probabilities (<= 0.5), overconfident for the probabilities between 0.5 and 0.8, and approaches perfect calibration for the high probabilities (> 0.8). For the $DistMult$ model, the calibration line is nearly perfect for all the probabilities. The interpretation plots clearly show that, in most cases, the relations significantly influence the models, a conclusion further reinforced by the magnitude of impact for each of the five triples. For the $TransE$, $TransH$, and $DistMult$

models, there are some instances where the greatest impact comes from the subject of the triples.

Looking at the predicted triple statistics, we can see that the models predict only three to four distinct relations. Many of them also have high probabilities, above 0.9. The $TransH$ model has the most triples (50), with a probability over 0.9. In terms of triples, the $ComplEx$ model has the highest average probability (0.469) and the lowest standard deviation (0.11). The models predict the following relations: $level$, $hasReference$, $term$, and $22 - rdf - syntax - ns\#type$. In this group, according to Table 6.17, the average training time for the models is 1200 seconds per epoch and 5 epochs, respectively.

In comparison to Group 4, the results in Group 5, shown in Tables 6.18, 6.19, and 6.20, which utilize the $GATLayer$, are marginally inferior. In this context, the $TransH$ for thresholds of 0.5 and 0.6 and the $HoIE$ for thresholds of 0.7 are the most effective models. In the clustering evaluation, the $DistMult$ model outperforms the others. Overall, the $RotatE$ model demonstrated subpar performance for this group. We can observe that we have high precision and low recall, or the opposite. The $DistMult$-like embeddings model's metrics were consistent across all three thresholds we used. It has an absolute precision of 1.0, but its recall is extremely low (0.02). All models in this category exhibit a greater tendency to overfit than those in Group 4. The precision-recall curves are largely consistent with the previous ones, with the exception of the $DistMult$ and $RotatE$-like embeddings models, which yield suboptimal results for this group. The probability calibration diagrams indicate that the models are overconfident for probabilities up to 0.7, while they are sufficiently close to the precisely calibrated line for higher probabilities. The relation appears to have a significant impact on the interpretation plots in the majority of cases; however, there are instances in which the subject or object had a greater impact.

Upon examining the predicted relation statistics, it is evident that there are no triples or relations with probabilities exceeding 0.9. The $TransE$ model has the highest number of triples (140), with a probability at or above 0.6 and a maximal predicted probability of 0.74. The results of the $HoIE$ model were also comparable. We can conclude that the $GATLayer$ does not appear to enhance the models. In comparison to the preceding groups, this layer also increases the training duration and the number of epochs that the models require to train. According to Table 6.21, 7 epochs and 3200 seconds per epoch are the average training times for the models, respectively.

Within Group 6, we make use of both the $GCNLayer$ and the $GATLayer$. After reviewing the metrics, shown in Tables 6.22, 6.23, and 6.24, it is evident that the models exhibit either high accuracy and poor recall, or vice versa for the higher thresholds. Furthermore, using both $GCNLayer$ and $GATLayer$ together enhances performance compared to using them separately. The $HoIE$ and $TransE$ embeddings models consistently outperformed other models across all three thresholds. The $HoIE$ model exhibited superior performance in the clustering assessment. The $DistMult$ model has a very high recall rate across all thresholds, although its accuracy value is lower. The training and validation charts once again demonstrate the indispensability of the dropout layers. The precision-recall curves of the previous groups' models are mostly consistent, except for the $DistMult$ and $ComplEx$ models, which perform somewhat worse. Upon examination of the probability calibration, it

is evident that the $TransE$, $TransH$, and $HoIE$ models have similar characteristics in their calibration. Models exhibit overconfidence for probabilities less than 0.5 and underconfidence for probabilities greater than 0.5.

The $DistMult$ model exhibits overconfidence over its entire range of probabilities, except for extremely high probabilities when the model demonstrates perfect calibration. The $ComplEx$ model only considers probabilities with values equal to or greater than 0.5. The model exhibits overconfidence for probabilities ranging from 0.5 to 0.8 and underconfidence for probabilities beyond this range. The interpretation plots exhibit a high degree of similarity to the preceding ones. Upon examining the statistics for the projected triples, it is evident that once again, the models do not forecast relations with a probability above 0.9. The $TransE$ model yields the highest probability, reaching a value of 0.74. This model forecasts 11 unique relationships. The $ComplEx$ model predicts 43 unique relations, with a maximum probability of 0.66 and a low standard deviation of 0.02. Using both layers, we reduced training time compared to Group 6. According to Table 6.25, the average training time is 6 epochs, or 2200 seconds per epoch.

In Group 7, we use the Conv1D architecture. For all thresholds, as shown in Tables 6.26, 6.27, and 6.28, the $NoEmbds$ model outperforms the other models. This model also performs better in the clustering evaluation. Overall, the $TransH$ and $ComplEx$ embedding models perform poorly. The training and validation curves indicate that the models don't overfit to an excessive degree. The precision-recall curve from the $TransE$ model, which performed well, is better than the other models. The curves from the $RotatE$ and $DistMult$ models are not excellent at all. For these models, the probability calibration plots also show that for probabilities up to 0.5, the models are overconfident. For probabilities above 0.5, the $DistMult$ model is overconfident, and the $RotatE$ model approaches perfect calibration. The interpretation plots again indicate that the relationship has a greater impact in most cases.

When examining the statistics from the predicted triples, we can see that all the models except $TransH$, which predicts only one distinct relation, predict a variety of relationships. The $NoEmbds$ model predicts 18 distinct relations with 9 of them having a probability above 0.9 and a maximum probability of 0.997. In this architecture, we have better precision roc-auc and pr-auc values across the thresholds, but we take lower recall and F1 scores. The models that used the Conv1D architecture took a much longer time per epoch to train than the models that used the Dense architecture. According to Table 6.29, the average training time is 9000 seconds per epoch, with a total of 7 epochs. Exceptions are the $DistMult$ and $RotatE$ models, which required much less time—about 1000 seconds per epoch—and had poor results for both of these models.

Within Group 8, we experiment with integrating the $GCNLayer$ and Conv1D architectures. It is evident, according to Tables 6.30, 6.31, and 6.32, that the accuracy improves but the recall and F1 score decrease. When comparing the models in Group 4, which used the Dense architecture with the same layer, we see significantly improved outcomes for all models and thresholds. It is worth mentioning that, once again, the model that consistently performed better than the others in most of the measures is the one that does not use sophisticated embeddings. Early stopping prevents overfitting by ending the training process at the right moment. The $NoEmbds$ model has a superior precision-recall curve, while the $TransE$ model also demonstrates commendable performance in terms of metrics, with a similarly impressive

precision-recall curve. Once again, the $TransH$ and $RotatE$ models exhibit unsatisfactory performance. The probability calibration plots demonstrate that the well-performing models exhibit accurately calibrated probabilities over the entire probability spectrum. The calibration becomes almost flawless, particularly for probabilities over 0.8.

By analyzing the statistics of the projected triples, it is evident that both the $HoIE$ and $DistMult$ models make many predictions of relations with significant probability, above 0.9. However, it is important to note that these models exhibit excessive confidence when it comes to high probabilities. The $TransE$ model, known for its accurate probability calibration plot, predicts 11 unique relationships, with two having a probability above 0.9 and the highest probability being 0.952. According to Table 6.33, the average training time is 12000 seconds per epoch, for a total of 7 epochs.

In Group 9, where we try the $GATLayer$ combined with the Conv1D architecture, we can see similar impacts to the models as in Group 5. We have observed in Tables 6.34, 6.35, and 6.36 an increase in precision surpassing that of Group 8, as well as a slight decrease in the remaining metrics. For this group, the $HoIE$ model outperforms the rest of the models, with the $NoEmbds$ model also performing very well. In the clustering evaluation, these two models performed the best. All of these models' evaluation plots are consistent with the excellent model metrics. We can still see poor results from the $RotatE$ and $TransH$ models. According to the predicted triple statistics, high-probability relationships exist only in the $DistMult$ and $TransE$ models. But these models don't have a decent probability calibration plot. The $NoEmbds$ model, which performed well, predicts 7 distinct relations with a maximum probability of 0.84. The $HoIE$ model, the best in this group, predicts 16 distinct relations with a maximum probability of 0.66. In terms of training time, we can see an increase in all models per epoch when adding $GATLayers$ compared with $GCNLayers$. According to Table 6.37, the average training time is 15000 seconds per epoch, with 7 epochs in total.

Finally, in Group 10, we utilize both $GCNLayer$ and $GATLayer$. The results shown in Tables 6.38, 6.39, and 6.40 again have high precision and lower recall values. These results don't appear to be significantly superior to using either the $GCNLayer$ or the $GATLayer$ alone, or when compared to Group 6, which utilizes both layers with Dense architecture. The lower recall values also affect the MCC value and the F1 score, resulting in better precision. The $NoEmbds$ and $HoIE$ models are superior to the others. The $HoIE$ also performs better in the clustering evaluation. Once again, we observe that the $RotatE$ model yields inferior results. The models that performed the best in this group also have excellent evaluation plots. The $NoEmbds$ model has a better precision-recall curve and almost perfect calibrated probabilities for the whole range of probabilities.

Also, for the majority of instances we examined, the interpretation plots show that the relation has the greatest impact on the model. There are also cases where other triple parts affect probabilities. The $HoIE$ model has overconfidently calibrated probabilities for the probabilities between 0.6 and 0.8 and approaches perfect calibration for the rest of the probabilities. In this model, the interpretation plots show that the subject has the greatest impact multiple times. Examining the statistics from the predicted triples, we can see that only the $HoIE$, $NoEmbds$, and $DistMult$ models predict relations with high probability, above 0.9. The $NoEmbds$ model predicts five distinct relationships, with three having a probability

above 0.9, a maximum probability of 0.94, and the lowest standard deviation having a value of 0.21.

The $HoIE$ model predicts 3 distinct relations, with one of them having a probability above 0.9, a maximum probability of 0.99, and a 0.26 standard deviation. The $DistMult$ model predicts 3 distinct relations, with 2 of them having a probability above 0.9, a maximum probability of 0.99, and a 0.32 standard deviation. Concerning the training time per epoch, the models required increased training time when adding both the $GCNLayer$ and $GATLayer$. The average training time, according to Table 6.41, is 20000 seconds per epoch, with 7 epochs in total.

Moving forward to the individual experiments, we can observe in Table 6.42 that the first experiment, using method 2 as a negative sampling generation strategy that multiplies the negative samples by 3, does not significantly improve the results. In contrast, we get poor results both for the triple classification and relation prediction tasks. According to Table 6.43, the training time for experiment 1 is 1550 seconds per epoch, and the required epochs are 5. When using method 3 as the negative sampling generation strategy in experiment 2, we can see in Table 6.44 far better results, and in some cases, even better results, compared with the models in groups 1–10. Precision is the only metric that is lower than the rest. The other metrics remain high across all three thresholds.

The model predicts 25 distinct relations with 21 of them having a probability above 0.9. The average probability is also higher than that of the other models, which typically range from 0.4 to 0.56. The maximum probability is 1, and the minimum is 0. Despite achieving excellent results, the clustering evaluation reveals poorer results compared to other models. This evaluation is merely a suggestion, and it is the sole method available to assess the newly suggested triples. According to Table 6.45, the training time for experiment 2 is 850 seconds per epoch, and the required epochs are 5.

The precision-recall curve is almost perfect; the training and validation plots show that the model doesn't overfit; and the interpretation plots show that the relation has the greatest impact on the probabilities. The probability calibration plot shows perfect calibration for the probabilities below 0.5, overconfidence for the probabilities between 0.5 and 0.9, and nearly perfect calibration for the rest of the probabilities. As shown in Tables 6.46 and 6.48, when we use method 2 as the negative sampling generation strategy in experiments 3 and 4, the models again give very poor results. According to Tables 6.47 and 6.49, the training time for experiment 3 is 1590 seconds per epoch with 5 epochs in total, and for experiment 4, it is 3100 seconds per epoch with 7 epochs in total.

When we use method 3 to obtain the negative samples in experiment 5, the model produces exceptional, nearly flawless results, as shown in Table 6.50. The precision-recall curve is nearly flawless, as are all of the metrics. The calibration plot indicates that the probabilities below 0.5 are overconfident, while the probabilities above 0.5 are underconfident. This was a rare occurrence in the other models. The model predicts 42 out of 43 relationships, with 40 of them having a probability greater than 0.9. The predicted triples have an exceptionally high average probability, with a value of 0.83. The utmost predicted probability is 0.99, and the

standard deviation is 0.12. According to Table 6.51, the training time for experiment 5 is 1500 seconds per epoch, with 13 epochs in total.

The interpretation plots show a significant impact on the subject, the relations, and the object. Also noteworthy is that the most frequent relation that this model predicts has a frequency of 134. That means the model doesn't focus too much on a single or a few relations. Despite still being subpar, the results of experiment 6 shown in Table 6.52, which uses method 2 to obtain negative samples, significantly improve over the other models that previously used this method. The model does not predict relationships with a high probability (> 0.9), and it has low recall and F1 score values. The model's probability calibration plot for the entire range of probabilities is also virtually flawless. According to Table 6.53, the training time for experiment 6 is 17400 seconds per epoch, with 7 epochs in total.

In experiments 7 and 8, we achieve exceptional results, as shown in Tables 6.54 and 6.56, by utilizing the Conv1D architecture in conjunction with method 3 to obtain the negative samples. Both investigations' precision-recall curves are perfect. In both experiments, the probability calibration plots are nearly identical, displaying underconfident probabilities for the entire range. In the seventh experiment, the model predicts 11 distinct relations, all of which have probabilities exceeding 0.9. In the eighth experiment, we observe five distinct relations yet again with extremely high probabilities. The average probability for both experiments is high, while the utmost predicted probability is 1. Additionally, both experiments share the most frequent relation, the $hasCode$, and several common relations in the top five. According to Tables 6.55 and 6.57, the training time for experiment 7 is 8400 seconds per epoch with 5 epochs in total, and for experiment 8, it is 20500 seconds per epoch with 5 epochs in total.

We still get poor results in experiment 9, as shown in Table 6.58, where we augment the dataset to ensure that positive triples remain positive and mark the rest as negative. The model forecasts only 4 relations, with a maximum predicted probability of a triple of 0.45. Furthermore, the training time per epoch increases, nearly doubling the time without augmentation. According to Table 6.59, the training time for experiment 9 is 3150 seconds per epoch, with 5 epochs in total. In experiment 10, combining the same augmentation method with the Conv1D architecture and the $GCNLayer$ significantly improves the model's performance, as Table 6.60 demonstrates. Although it still doesn’t predict relations with probabilities above 0.9. The model has an excellent precision-recall curve and a great probability calibration plot. The interpretation plots demonstrate that the relationship has a greater impact. Notably, despite the augmentation, according to Table 6.61, this model required 7900 seconds for training per epoch and 5 epochs in total, which is both less time and epochs than the model without the augmentation.

In experiment 11, we employ method 2 to obtain negative samples, a method that yielded poor results in previous experiments. However, our results, based on Table 6.62, are superior to those of previous experiments that employed this method. The recall and F1 score values for this experiment, conducted using method 2, are also smaller than the precision. The precision recall is good, and the probability calibration plot shows near-perfect calibration for the entire probability range. The interpretation plots show a significant impact on the probabilities for all parts of the triples. The model predicts 21 relations with only three of them

having a probability above 0.6. According to Table 6.63, the training time for experiment 11 is 15000 seconds per epoch, and the required epochs are 5.

In experiment 12, we use the same model as in experiment 11, but with method 3 for getting the negative samples, we can again see perfect results for all the thresholds we use in Table 6.64. The precision-recall curve is perfect, and the probability calibration plot shows very good, calibrated probabilities. The interpretation plots again show a significant impact on the probabilities for all parts of the triples. The model predicts 20 distinct relations with 17 of them having probabilities above 0.9. The predicted triples have an average probability of 0.3, a maximum probability of 1, and a standard deviation of 0.42. In terms of training time per epoch, we can see that in this experiment, it was 7400 seconds compared to 15000 without the augmentation method. However, according to Table 6.65, it required 13 epochs, compared to 7 epochs for the model without the augmentation method.

Table 6.66 demonstrates the unsatisfactory performance of the custom attention layer in experiment 13. The model utilizes all the specified epochs, and although we see a decrease in both training and validation losses, the obtained results are not satisfactory. According to Table 6.67, the training time for experiment 13 is 7800 seconds per epoch, and the required epochs are 15. We concluded experiment 14, where we implemented dynamic embeddings, which produced modest results in terms of metrics, as shown in Table 6.68. The model exhibits a steep and accurate precision-recall curve. The probability calibration figure shows underconfident probabilities below 0.5, overconfident probabilities ranging from 0.5 to 0.9, and a well-calibrated range for the remaining probabilities. The interpretation plots clearly demonstrate that the relationship has a substantial influence on the probability.

The model makes predictions for 8 unique relationships, with an average probability of 0.42 for the triples. The highest probability observed is 0.87, and the standard deviation is 0.25. The most frequently predicted relations by this model have significant differences in frequencies. The relation with the highest frequency is $databasePath$, which occurs 1124 times. The second most common relationship is $hasFrequentTerm$, which occurs 350 times. The third most frequent relation is $hasCode$, occurring 14 times. According to Table 6.69, the training time for experiment 14 is 30500 seconds per epoch, and the required epochs are 7.

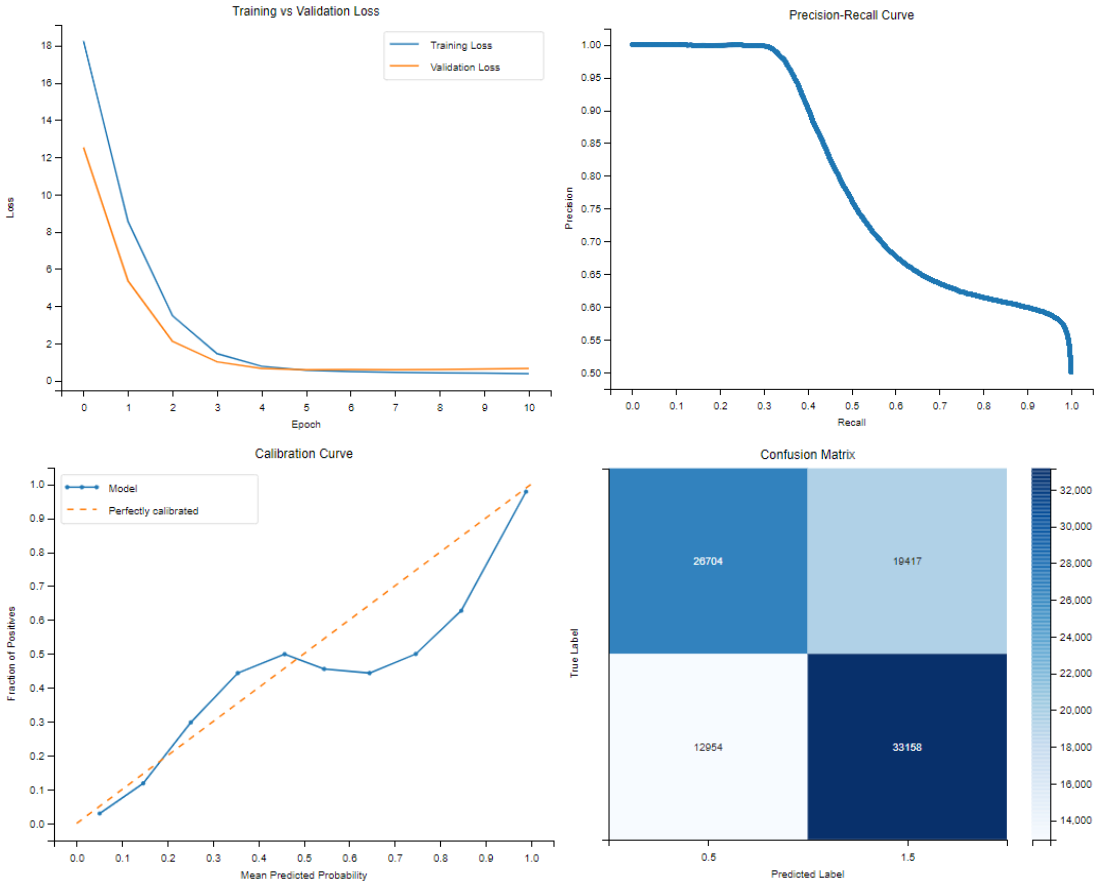
The number of epochs remains constant, but the training time per epoch is more than 5000 seconds. In general, all individual experiments utilizing method 2 for obtaining negative samples resulted in a significantly longer training time per epoch, which was expected given the higher number of negative samples compared to the other negative sampling generation strategies we employed. Method 3 sometimes slightly increases the training time per epoch. The epochs required for training remained almost the same for most experiments compared with the corresponding models from groups 1–10.

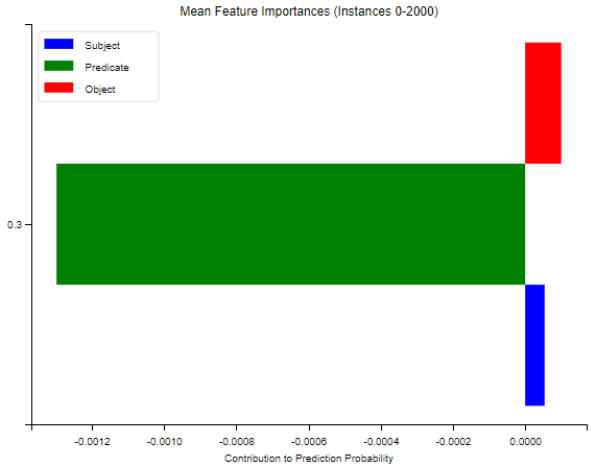
Detailed Experiments Results

I. Group 1

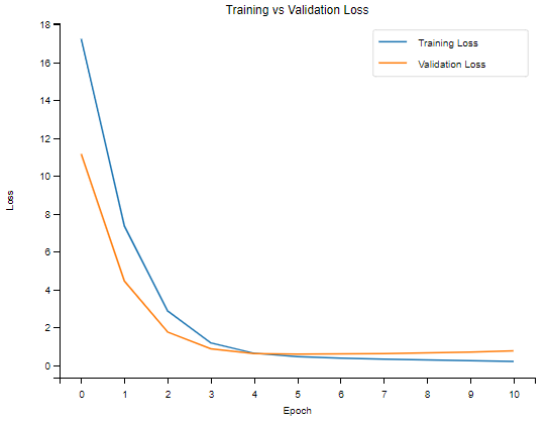
	Mean Silhouette	Mean Davies-Bouldin	Mean Calinski-Harabasz
$TransE$	0.8020	0.2544	3707.7435
$TransH$	0.8028	0.2540	3599.7435
$RotatE$	**0.8066**	**0.2497**	**4204.4276**
$HoIE$	0.8120	0.2425	3590.5345
$DistMult$	0.7868	0.2827	3123.9431
$ComplEx$	0.8059	0.2507	3958.5715

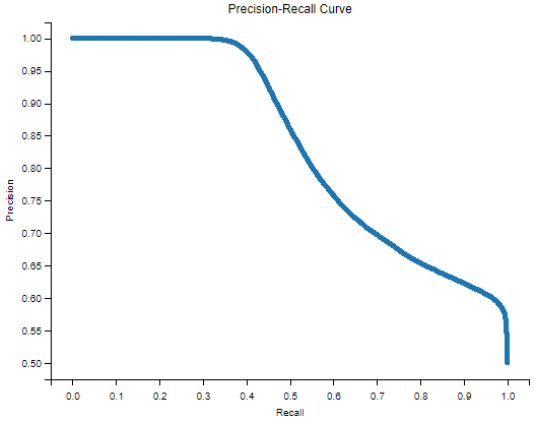
$TransE$

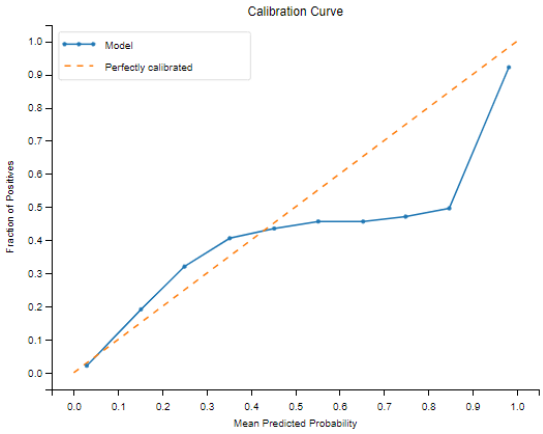


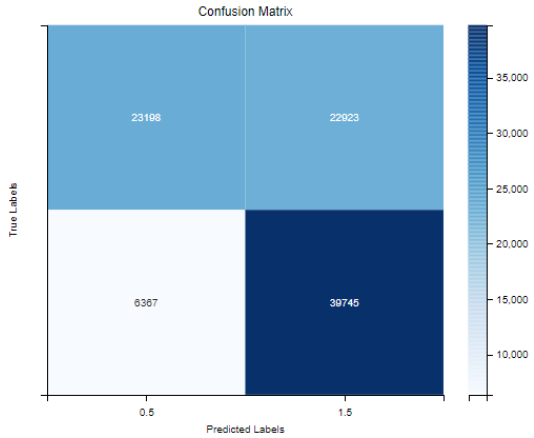

Mean Feature Importances (Instances 0-2000)
Subject
Predicate
Object
0.3
-0.0012
-0.0010
-0.0008
-0.0006
-0.0004
-0.0002
0.0000
Contribution to Prediction Probability

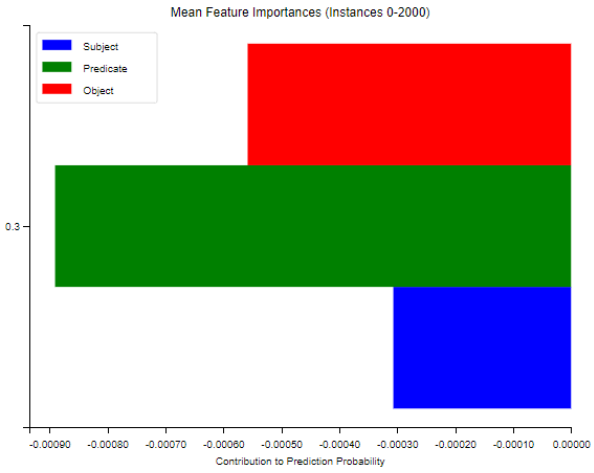
$TransH$


Training vs Validation Loss
Training Loss
Validation Loss
Loss
Epoch

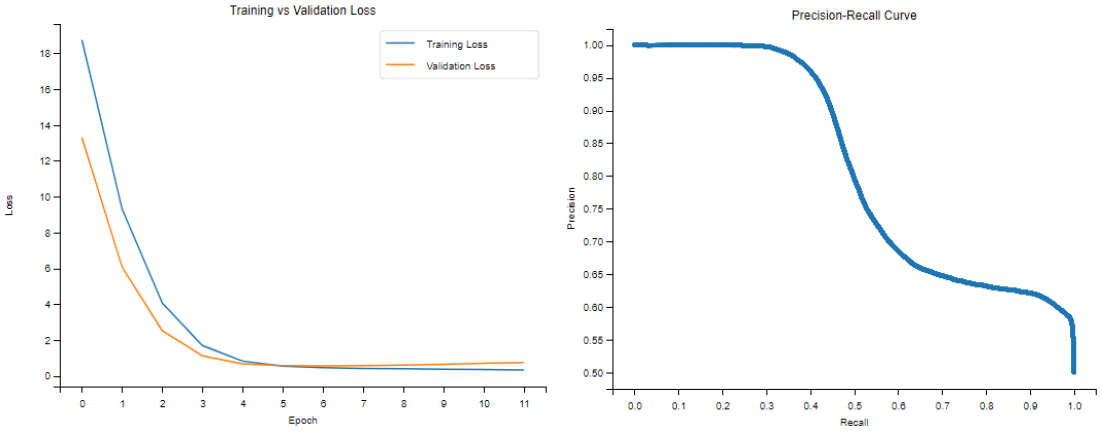

Precision-Recall Curve
Precision
Recall


Calibration Curve
Model
Perfectly calibrated
Fraction of Positives
Mean Predicted Probability

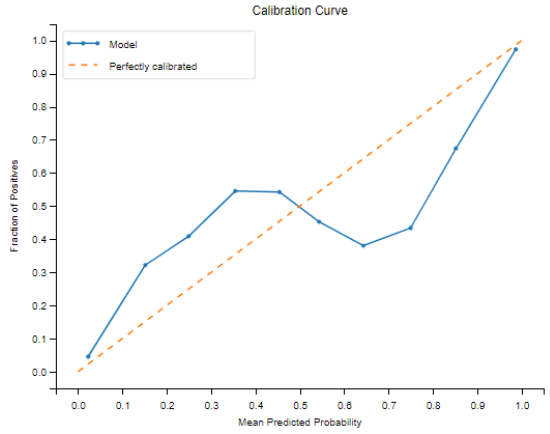

Confusion Matrix
23198
22923
6367
39745
True Labels
Predicted Labels

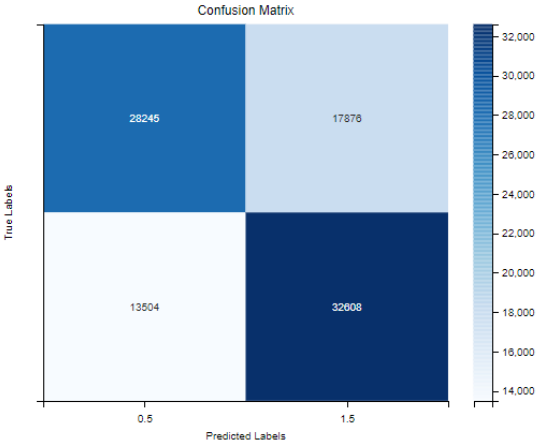

Mean Feature Importances (Instances 0-2000)
Subject
Predicate
Object
0.3
-0.00090
-0.00080
-0.00070
-0.00060
-0.00050
-0.00040
-0.00030
-0.00020
-0.00010
0.00000
Contribution to Prediction Probability

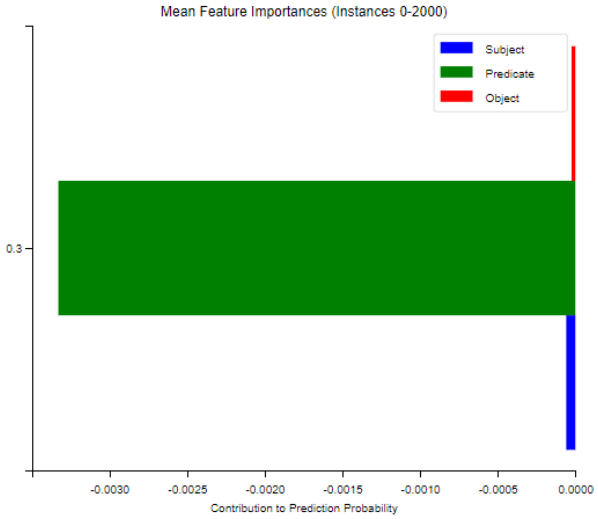
RotatE


Training vs Validation Loss
Training Loss
Validation Loss
Loss
Epoch

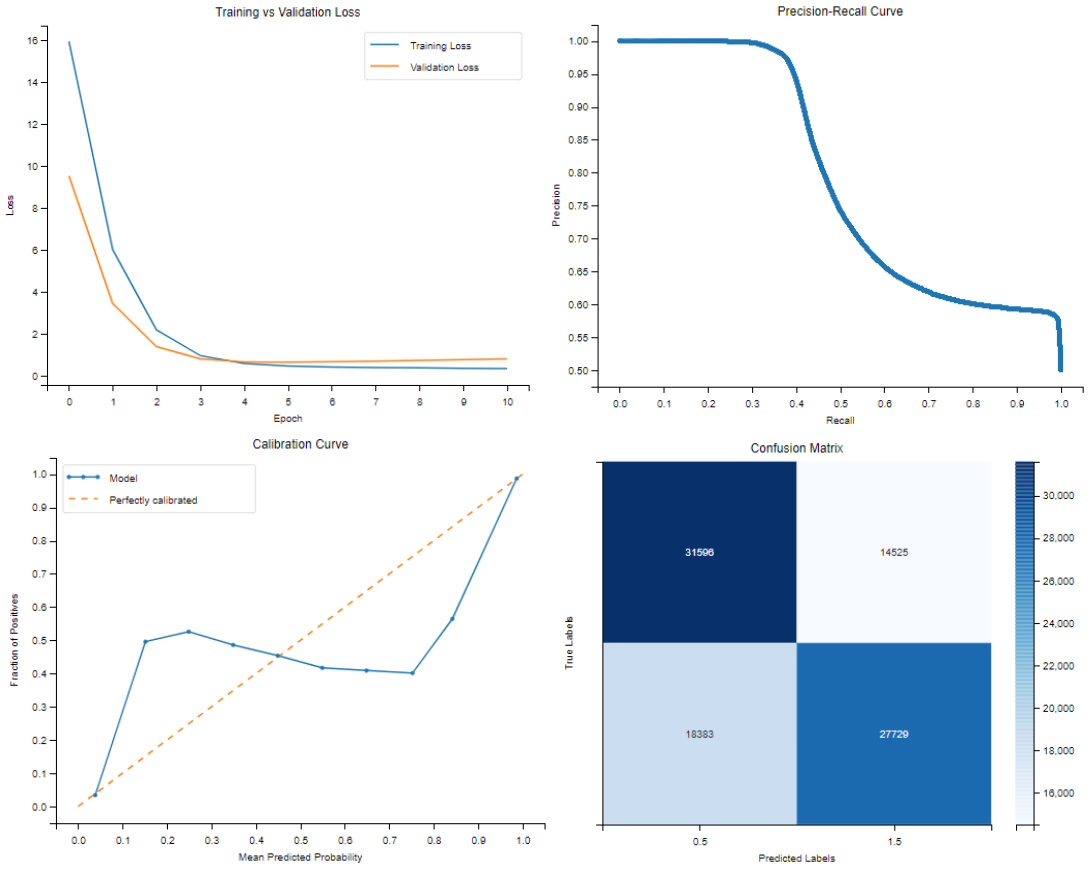
Precision-Recall Curve
Precision
Recall

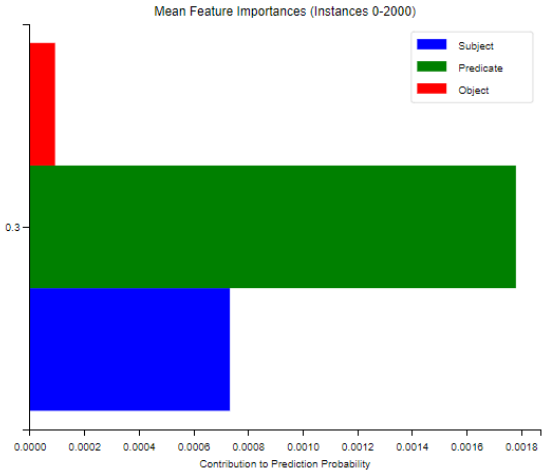

Calibration Curve
Model
Perfectly calibrated
Fraction of Positives
Mean Predicted Probability


Confusion Matrix
28245
17876
13504
32608
True Labels
Predicted Labels

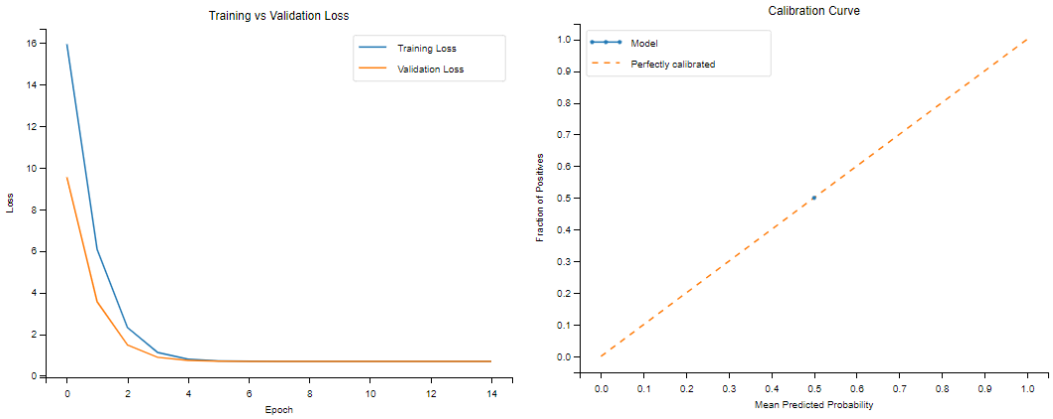

Mean Feature Importances (Instances 0-2000)
Subject
Predicate
Object
0.3
-0.0030
-0.0025
-0.0020
-0.0015
-0.0010
-0.0005
0.0000
Contribution to Prediction Probability

HoIE

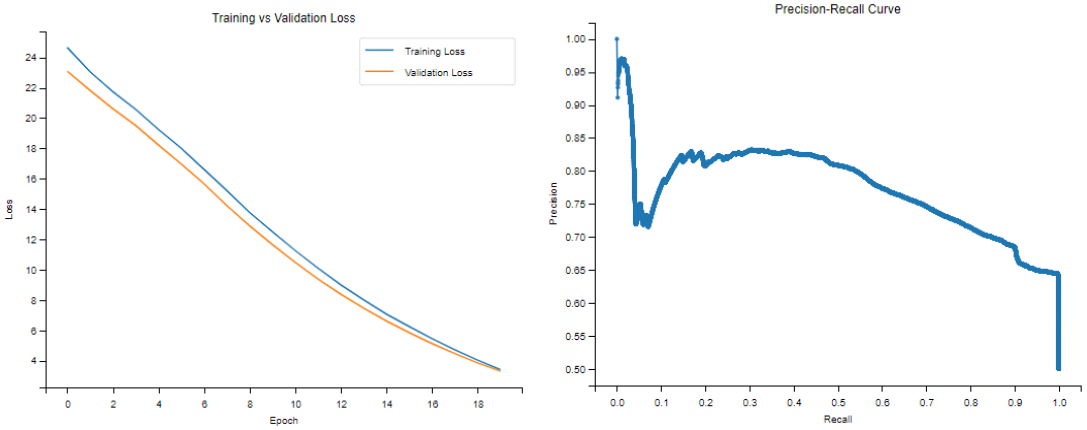

Training vs Validation Loss
Training Loss
Validation Loss
Loss
Epoch
Precision-Recall Curve
Precision
Recall
Calibration Curve
Model
Perfectly calibrated
Fraction of Positives
Mean Predicted Probability
Confusion Matrix
31596
14525
18383
27729
True Labels
Predicted Labels

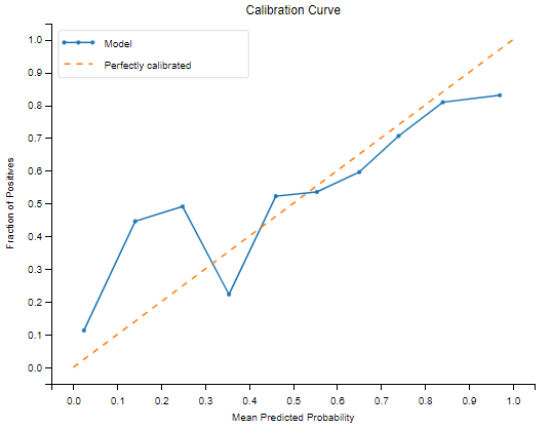


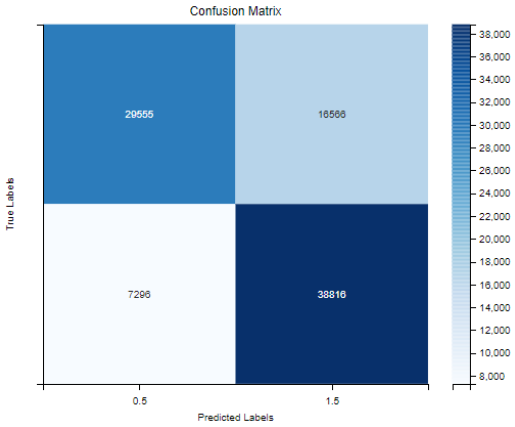
DistMult

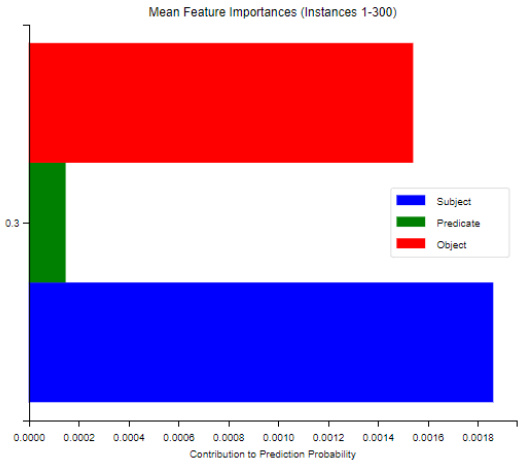


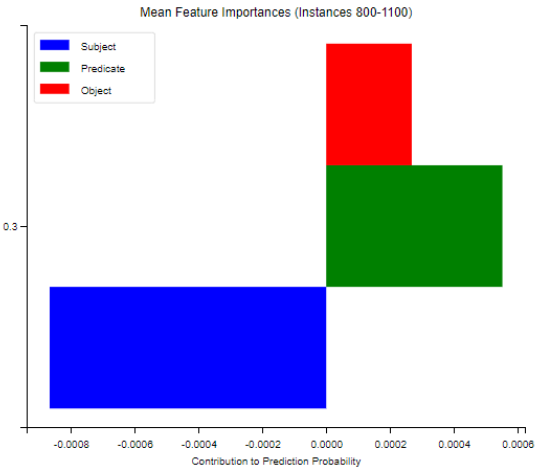
ComplEx

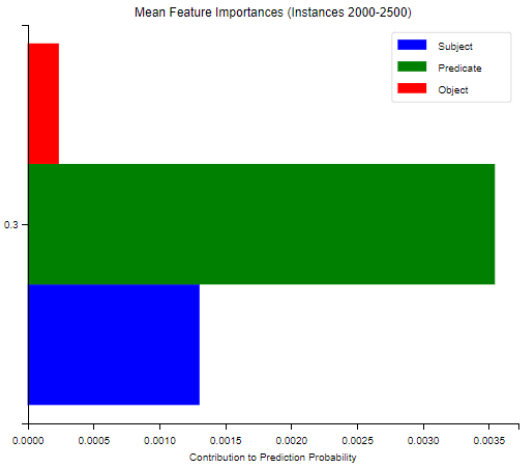












	subject	predicate	object
Triple 1	-0.1024	-0.1435	-0.0821
Triple 2	0.3715	-0.1040	-0.0422
Triple 3	-0.1175	-0.1458	0.2231
Triple 4	-0.1051	-0.1106	-0.0583
Triple 5	-0.1177	0.3810	-0.0701

	C1	C2	C3	C4	C5	C6	C7	C8
$TransE$	9	375	61	7	0.4642	0.9959	0.0077	0.2282
$TransH$	29	698	118	16	0.5148	0.9999	0.0001	0.3045
$RotatE$	22	291	11	5	0.3690	0.9478	0.0002	0.2509
$HoIE$	5	291	0	0	0.3333	0.8631	0.0028	0.2605
$DistMult$	1	0	0	0	0.5004	0.5004	0.5004	0
$ComplEx$	13	281	2	2	0.3154	0.9057	0.0001	0.2514

$TransE$

1. **Top 5 most frequent relations**:
   [('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/term', 436),
   ('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/context', 402),
   ('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasCode', 374),
   ('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/keyword', 121),
   ('http://www.w3.org/1999/02/22-rdf-syntax-ns#rest', 98)]
2. **Relations appearing only once**:
   ['https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasReference']
3. **Relation with highest average probability**:
   ('http://www.w3.org/2000/01/rdf-schema#subClassOf', 0.5279035317046302)
4. **Relation with lowest average probability**:
   ('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasReference',
   0.38319316506385803)

$TransH$

1. **Top 5 most frequent relations**:
   [('http://www.w3.org/2002/07/owl#imports', 852),
   ('http://www.w3.org/2002/07/owl#equivalentClass', 374),
   ('http://www.w3.org/1999/02/22-rdf-syntax-ns#first', 40),
   ('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasURI', 22),
   ('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasSubTheme', 20)]
2. **Relations appearing only once**:
   ['https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasCode',
   'http://www.w3.org/2002/07/owl#equivalentProperty',
   'https://ec.europa.eu/eurostat/NLP4StatRef/ontology/definition']
3. **Relation with highest average probability**:
   ('http://www.w3.org/2000/01/rdf-schema#range', 0.7395107001066208)
4. **Relation with lowest average probability**:
   ('http://www.w3.org/2002/07/owl#equivalentProperty', 0.1863100230693817)

$RotatE$

1. **Top 5 most frequent relations**:
   [('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasReference', 364),

('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasURI', 192),
('http://www.w3.org/2000/01/rdf-schema#subClassOf', 167),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/definition', 128),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/remark', 124)]

2. **Relations appearing only once**:
['https://ec.europa.eu/eurostat/NLP4StatRef/ontology/dataSource',
'http://www.w3.org/2000/01/rdf-schema#label']
3. **Relation with highest average probability**:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/dataSource',
0.5118739008903503)
4. **Relation with lowest average probability**:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/keyword',
0.24597927373006792)

$HoIE$

1. **Top 5 most frequent relations**:
[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasReference', 1233),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasParagraph', 226),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/fileLink', 29),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/context', 11),
('http://www.w3.org/1999/02/22-rdf-syntax-ns#type', 1)]
2. **Relations appearing only once**:
['http://www.w3.org/1999/02/22-rdf-syntax-ns#type']
3. **Relation with highest average probability**:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/fileLink',
0.41237134858965874)
4. **Relation with lowest average probability**:
('http://www.w3.org/1999/02/22-rdf-syntax-ns#type', 0.1697981059551239)

$DistMult$

1. **Top 5 most frequent relations**:
[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/relatedTerm', 1500)]
2. **Relation with highest average probability**:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/relatedTerm',
0.5004311203956604)
3. **Relation with lowest average probability**:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/relatedTerm',
0.5004311203956604)

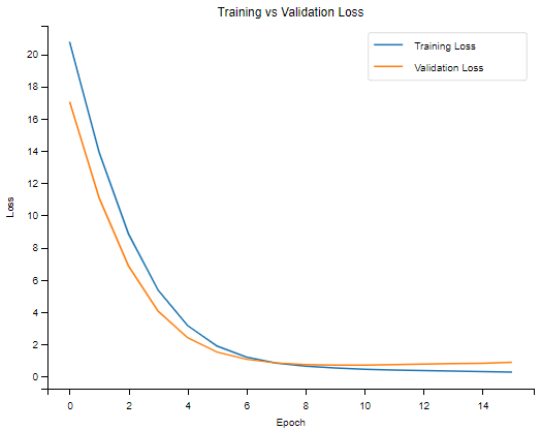
$ComplEx$

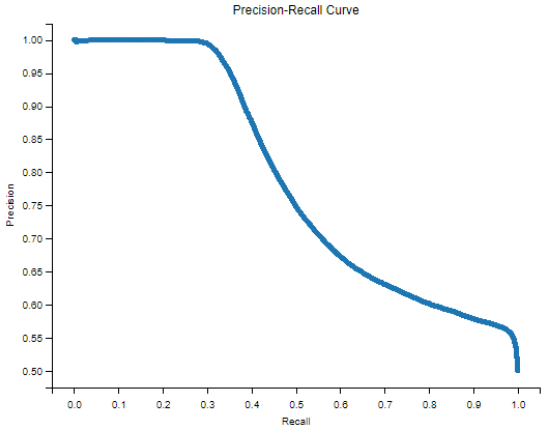
1. **Top 5 most frequent relations**:
   [('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasParagraph', 458),
   ('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/content', 427),
   ('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/fileLink', 244),
   ('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/title', 80),
   ('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasReference', 51)]
2. **Relations appearing only once**:
   ['https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasOECDTheme']
3. **Relation with highest average probability**:
   ('http://www.w3.org/2000/01/rdf-schema#label', 0.4366015885025263)
4. **Relation with lowest average probability**:
   ('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasOECDTheme',
   0.10760148614645004)

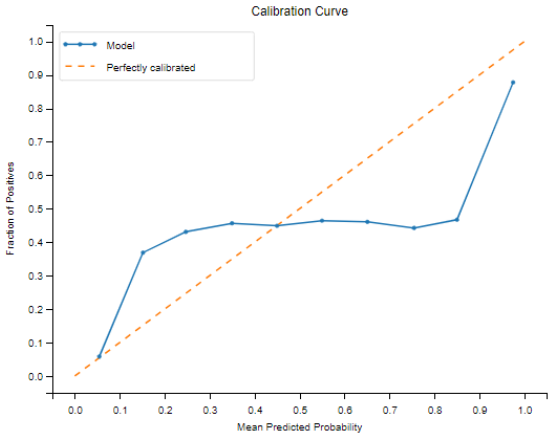
II. Group 2

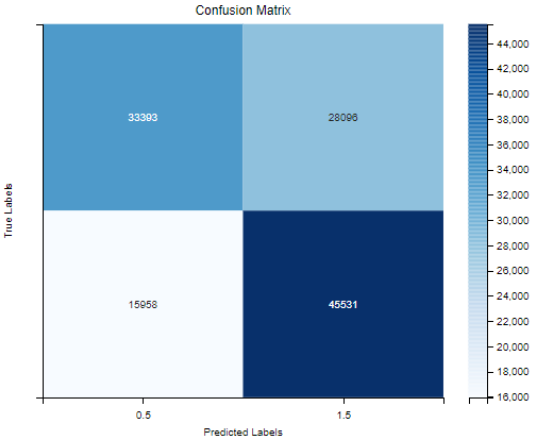
	Mean Silhouette	**Mean Davies-Bouldin**	**Mean Calinski-Harabasz**
$TransE$	0.8095	0.2452	3767.3763
$TransH$	0.8076	0.2466	3366.1078
$RotatE$	0.8057	0.2484	3442.7084
$HoIE$	0.8015	0.2531	3332.1427
$DistMult$	0.7867	0.2825	3122.6344
$ComplEx$	**0.8137**	**0.2384**	**3917.0213**

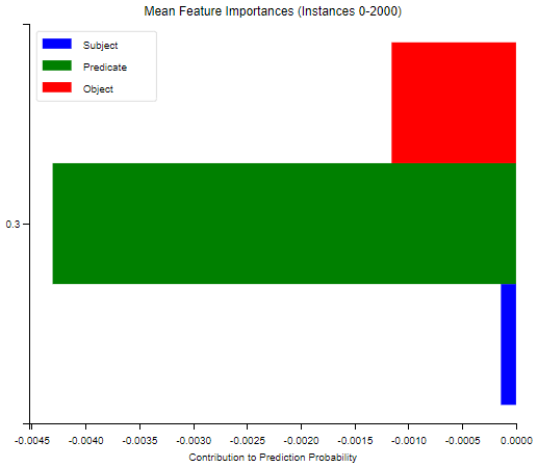
$TransE$



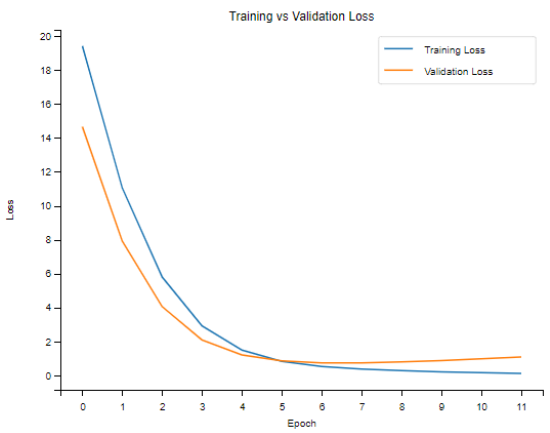


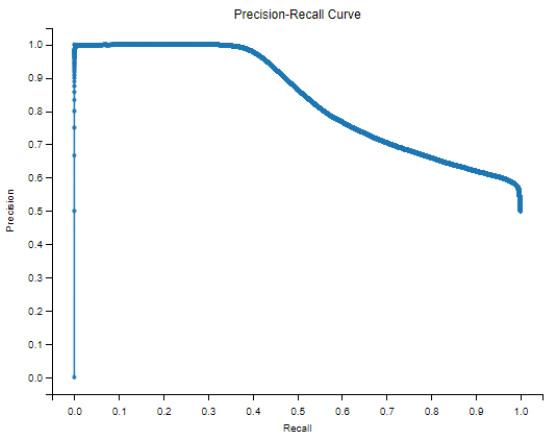

Calibration Curve
Model
Perfectly calibrated
Fraction of Positives
Mean Predicted Probability

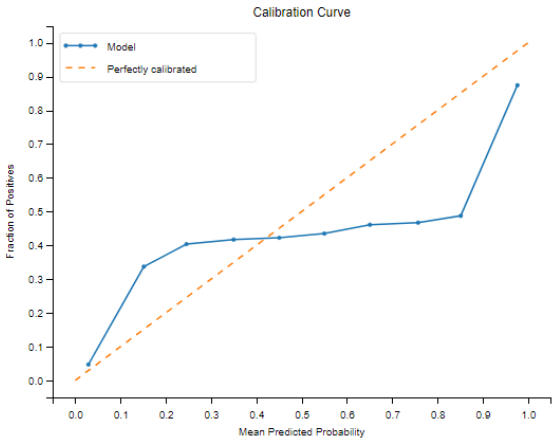

Confusion Matrix
33393
28096
15958
45531
True Labels
Predicted Labels

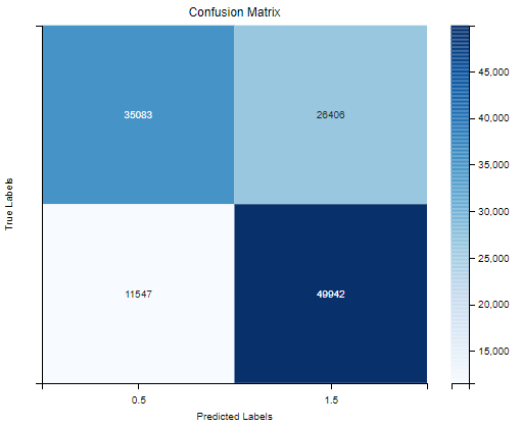

Mean Feature Importances (Instances 0-2000)
Subject
Predicate
Object
Contribution to Prediction Probability

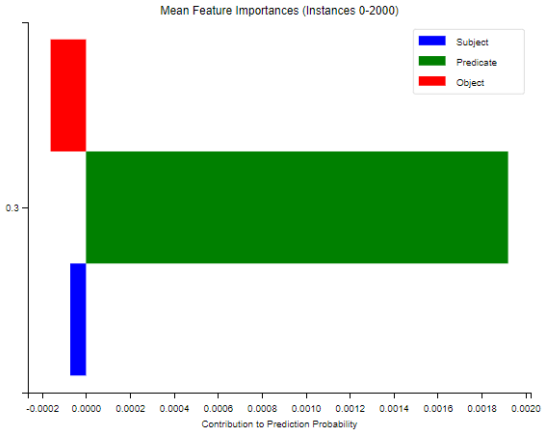
$TransH$


Training vs Validation Loss
Training Loss
Validation Loss
Loss
Epoch

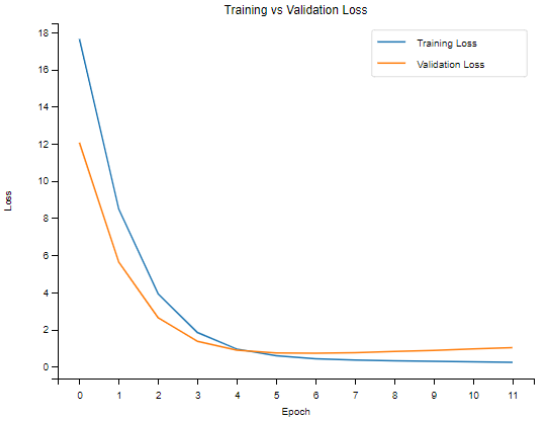

Precision-Recall Curve
Precision
Recall

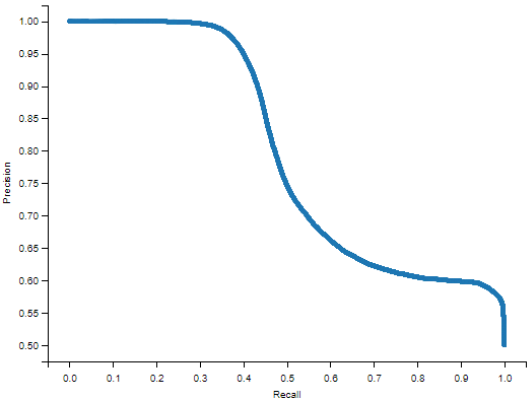

Calibration Curve
Model
Perfectly calibrated
Fraction of Positives
Mean Predicted Probability

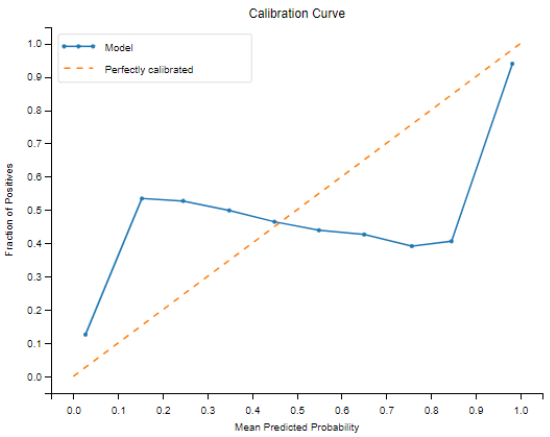

Confusion Matrix
35083
26406
11547
49942
True Labels
Predicted Labels

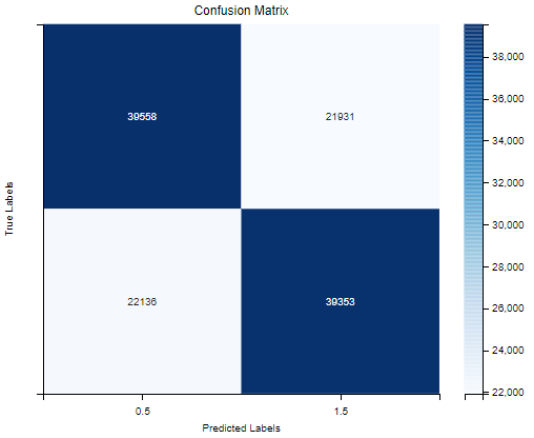

Mean Feature Importances (Instances 0-2000)
Subject
Predicate
Object
Contribution to Prediction Probability

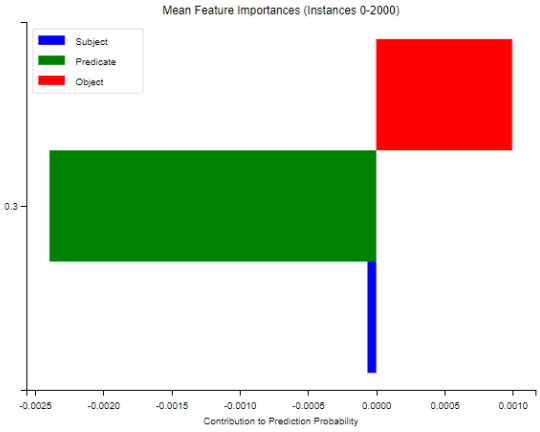
RotatE


Training vs Validation Loss
Training Loss
Validation Loss
Loss
Epoch


Precision-Recall Curve
Precision
Recall

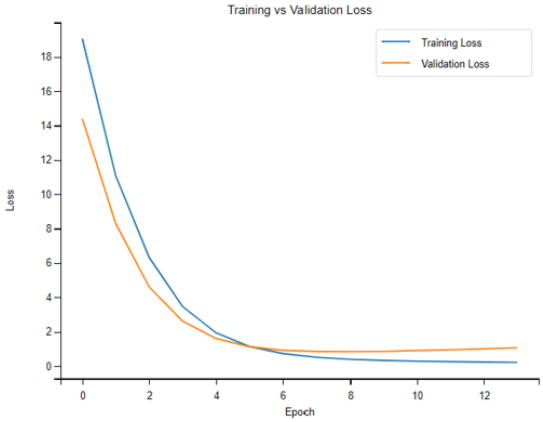


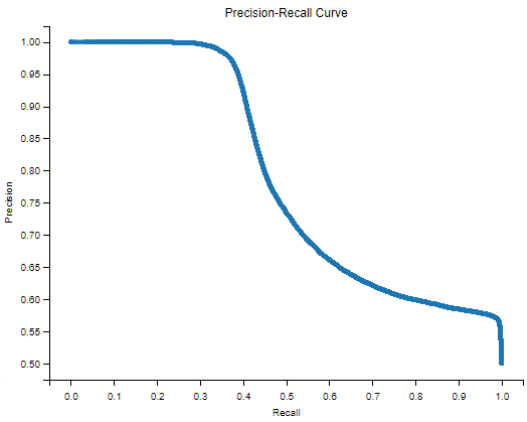


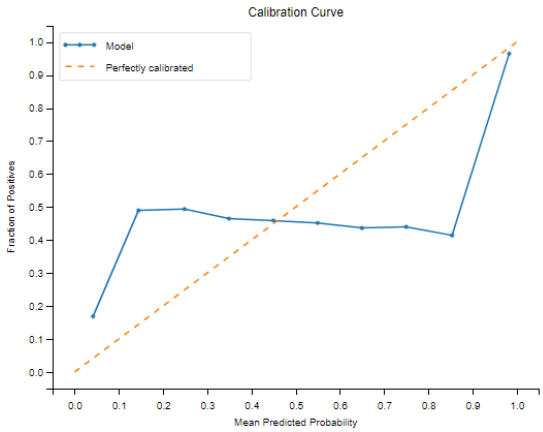


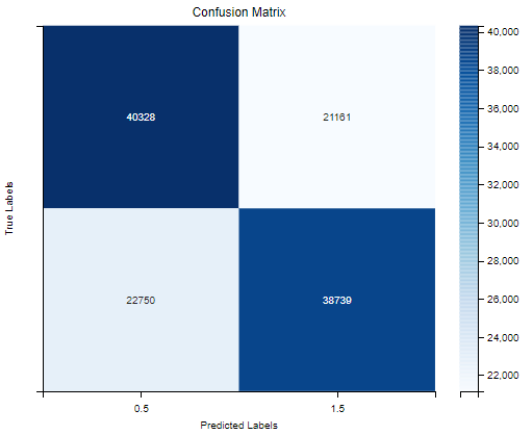
	subject	predicate	object
Triple 1	0.0008	-0.23360	-0.0828
Triple 2	-0.0086	0.04165	-0.0849
Triple 3	-0.0294	0.22383	-0.0857
Triple 4	0.0194	0.24671	0.0156
Triple 5	0.0257	0.09105	0.2120

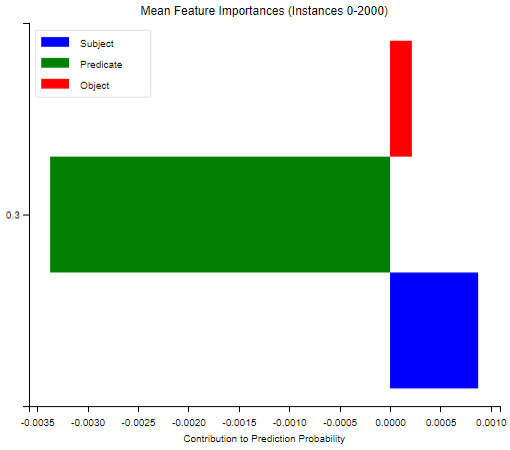
$HoIE$



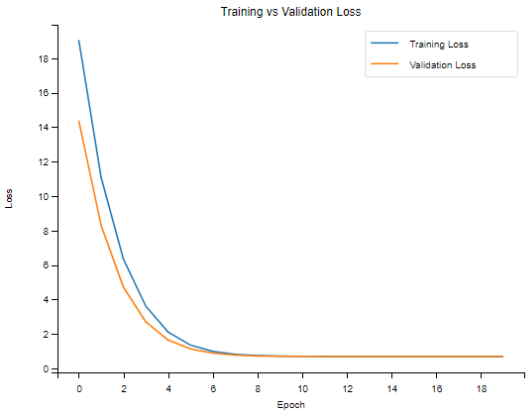


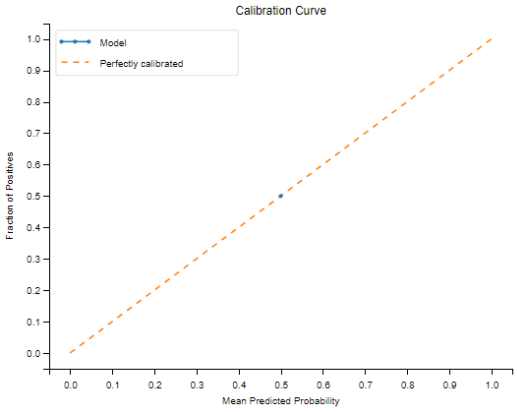

Calibration Curve
Model
Perfectly calibrated
Fraction of Positives
Mean Predicted Probability


Confusion Matrix
40328
21161
22750
38739
True Labels
Predicted Labels

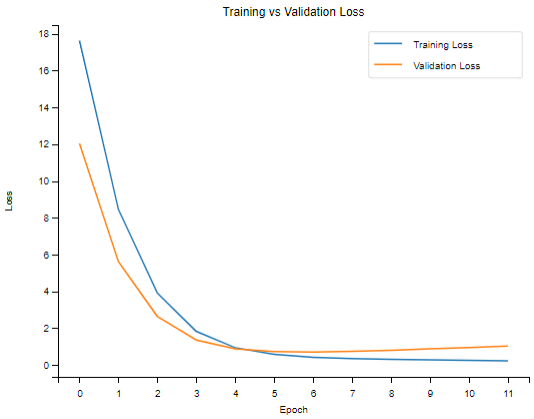

Mean Feature Importances (Instances 0-2000)
Subject
Predicate
Object
Contribution to Prediction Probability

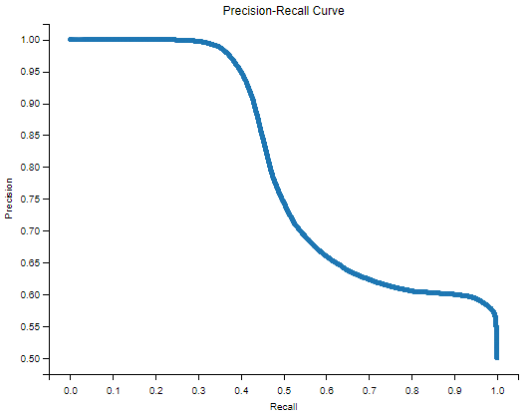
DistMult

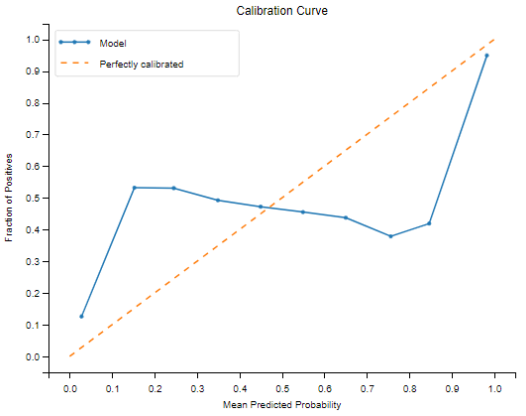

Training vs Validation Loss
Training Loss
Validation Loss
Loss
Epoch

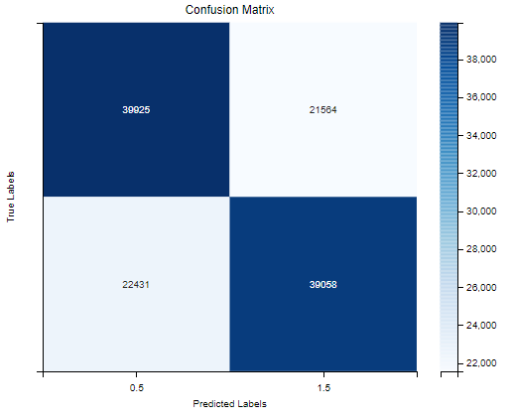

Calibration Curve
Model
Perfectly calibrated
Fraction of Positives
Mean Predicted Probability

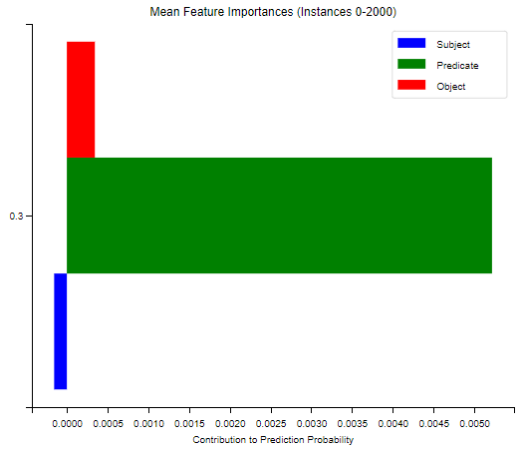
$ComplEx$











	C1	C2	C3	C4	C5	C6	C7	C8
$TransE$	4	573	42	3	0.4451	0.9463	0.0002	0.3022
$TransH$	1	638	210	1	0.4824	1.0000	0.0001	0.3560
$RotatE$	4	298	25	2	0.2966	0.9991	0.0001	0.2928
$HoIE$	1	355	7	1	0.3524	0.9153	0.0009	0.2804
$DistMult$	1	0	0	0	0.5	0.5	0.5	0
$ComplEx$	14	484	48	6	0.4121	0.9856	0.0001	0.2973

$TransE$

Top 5 most frequent relations:
[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasReference', 796),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasParagraph', 674),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/content', 26),
('http://www.w3.org/2002/07/owl#unionOf', 4)]
Relation with highest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasReference', 0.447425178314189)
Relation with lowest average probability:
('http://www.w3.org/2002/07/owl#unionOf', 0.4008448729291558)

$TransH$

Top 5 most frequent relations:
[('http://www.w3.org/2002/07/owl#equivalentClass', 1421)]
Relation with highest average probability:
('http://www.w3.org/2002/07/owl#equivalentClass', 0.4828190933904544)
Relation with lowest average probability:
('http://www.w3.org/2002/07/owl#equivalentClass', 0.4828190933904544)

$RotatE$

Top 5 most frequent relations:
[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/term', 1253),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/level', 204),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasReference', 3),
('http://www.w3.org/1999/02/22-rdf-syntax-ns#type', 3)]
Relation with highest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasReference',
0.4204809255897999)
Relation with lowest average probability:
('http://www.w3.org/1999/02/22-rdf-syntax-ns#type', 0.13876665631930032)

$HoIE$

Top 5 most frequent relations:
[('http://www.w3.org/2002/07/owl#versionInfo', 1500)]
Relation with highest average probability:
('http://www.w3.org/2002/07/owl#versionInfo', 0.3523507455016952)
Relation with lowest average probability:
('http://www.w3.org/2002/07/owl#versionInfo', 0.3523507455016952)

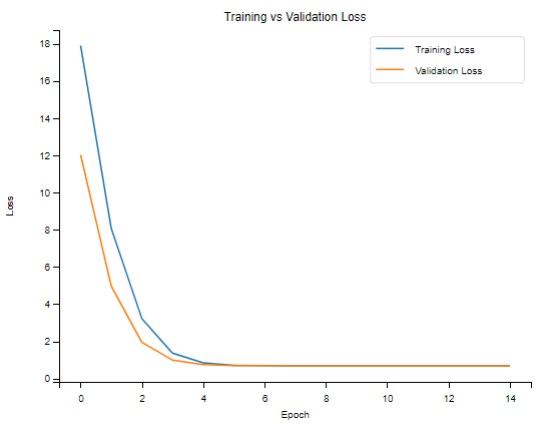
$ComplEx$

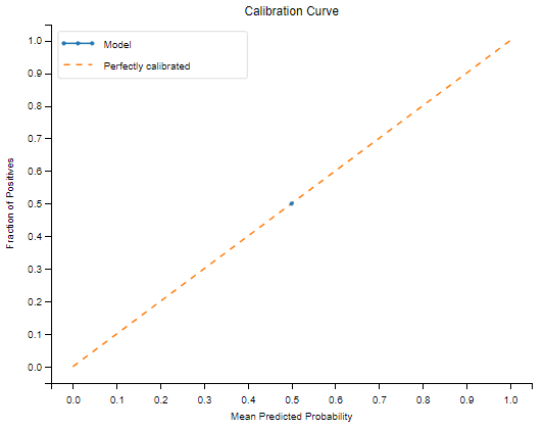
Top 5 most frequent relations:
[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/definition', 862),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/title', 216),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasReference', 145),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/id', 85),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasURL', 69)]
Relations appearing only once:
['https://ec.europa.eu/eurostat/NLP4StatRef/ontology/content']
Relation with highest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/content', 0.571976363658905)
Relation with lowest average probability:
('http://www.w3.org/1999/02/22-rdf-syntax-ns#type', 0.2250628693960607)

III. Group 3

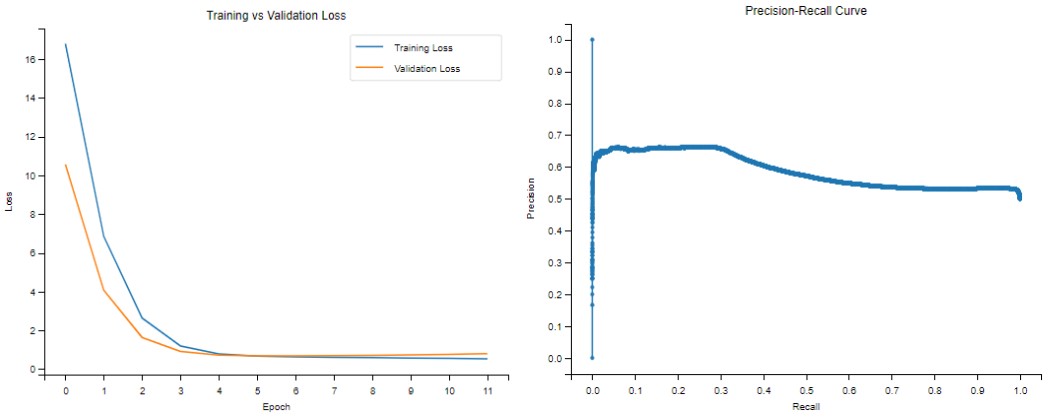
	Mean Silhouette	**Mean Davies-Bouldin**	**Mean Calinski-Harabasz**
$TransE$	0.7858	0.2831	3103.4511
$TransH$	0.7958	0.2625	3416.8373
$RotatE$	0.8122	0.2410	3699.4227
$HoIE$	0.8115	0.2410	4423.7205
$DistMult$	**0.8144**	**0.2384**	**4125.0630**
$ComplEx$	0.7833	0.2871	3062.0203

$TransE$

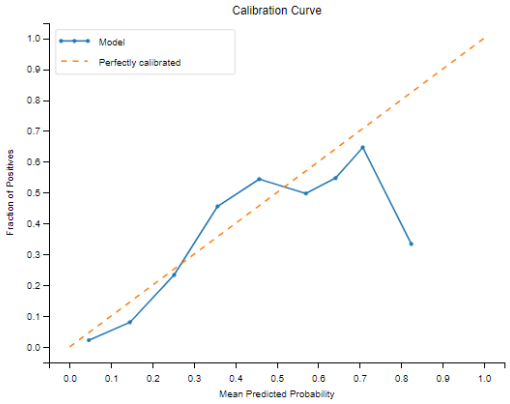


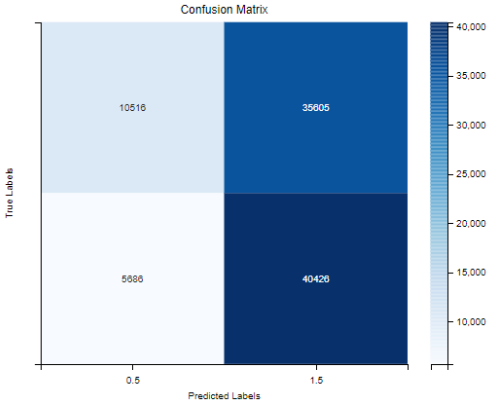


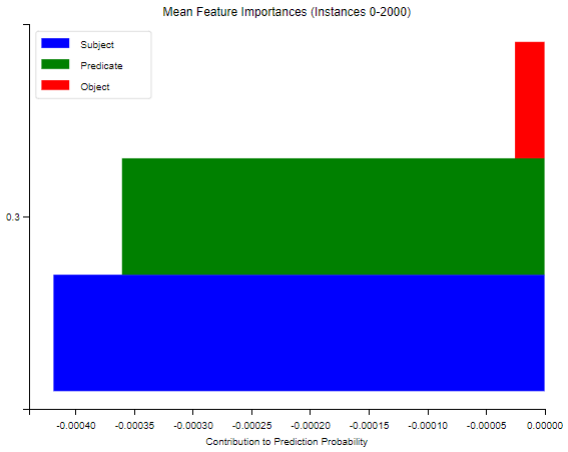
$TransH$


Training vs Validation Loss
Training Loss
Validation Loss
Loss
Epoch

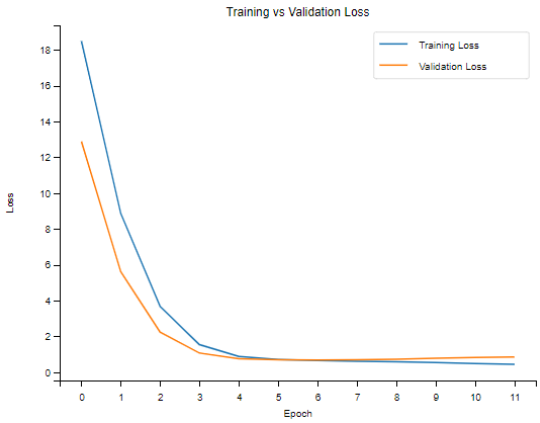
Precision-Recall Curve
Precision
Recall

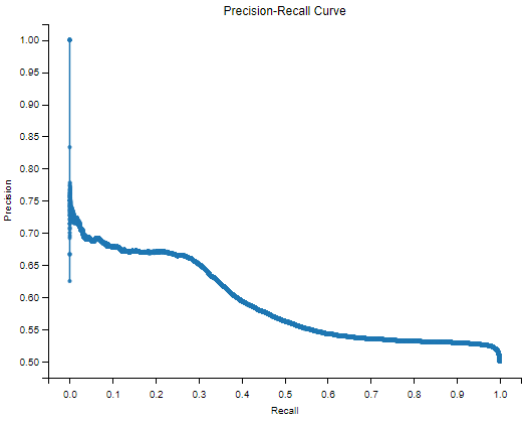

Calibration Curve
Model
Perfectly calibrated
Fraction of Positives
Mean Predicted Probability

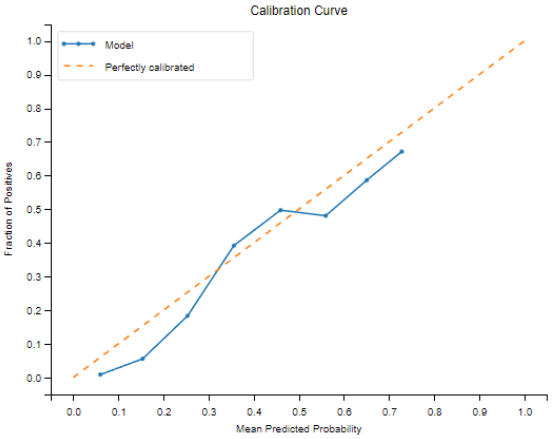

Confusion Matrix
10516
35605
5686
40426
True Labels
Predicted Labels

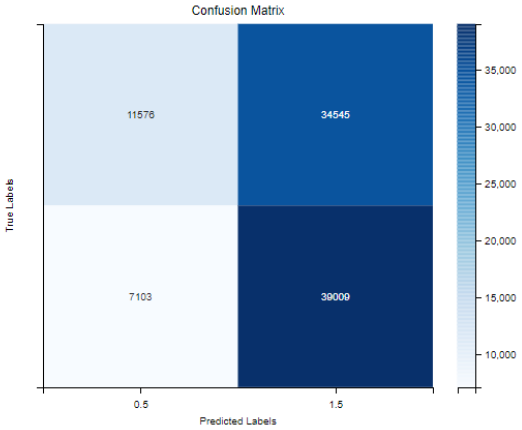

Mean Feature Importances (Instances 0-2000)
Subject
Predicate
Object
Contribution to Prediction Probability

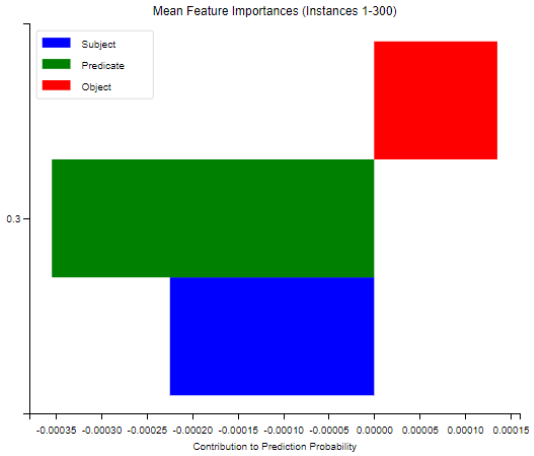
$RotatE$

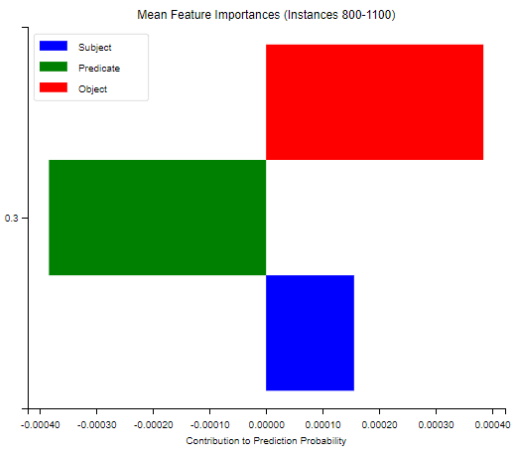


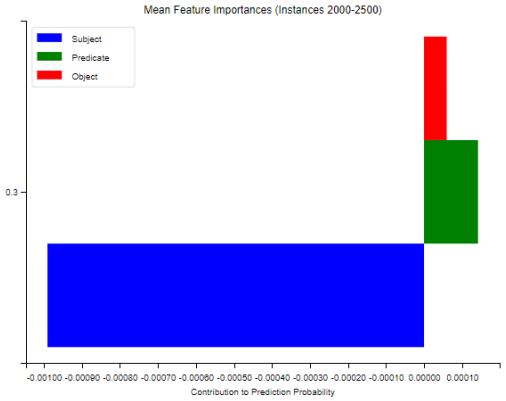






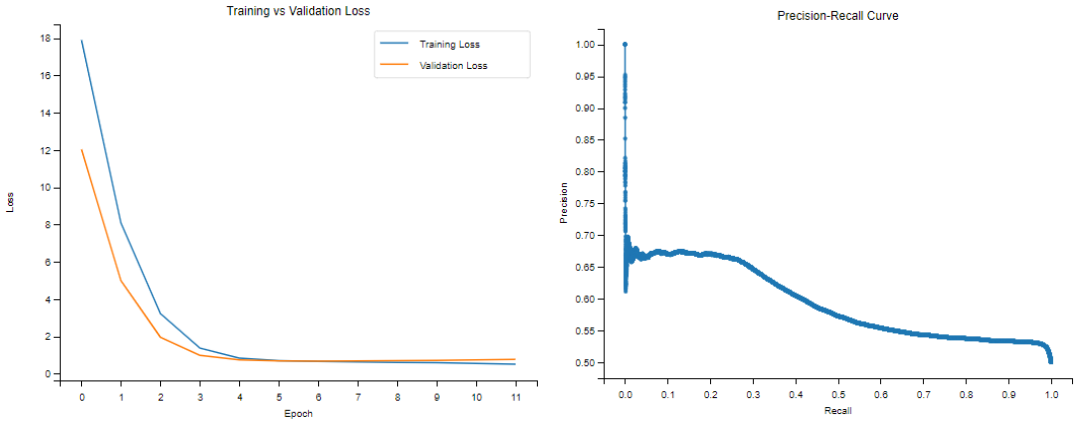




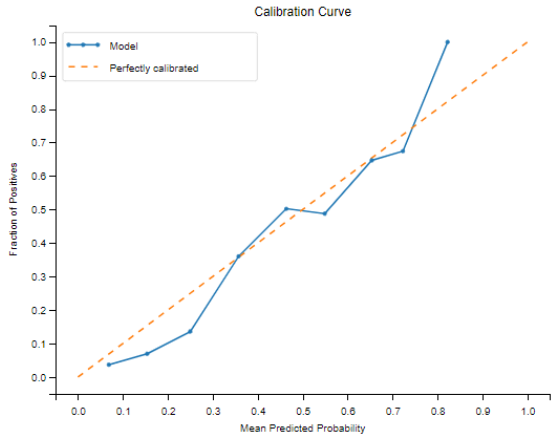


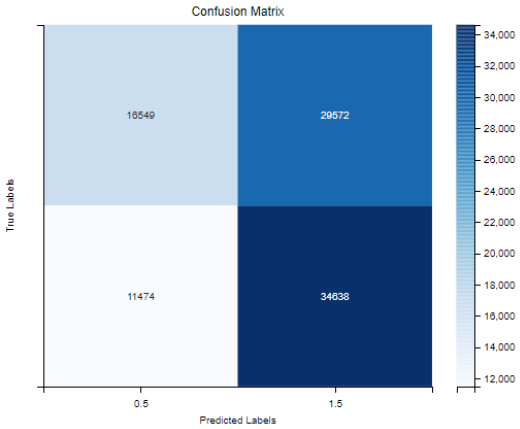
	subject	predicate	object
Triple 1	0.0153	0.2113	0.0424
Triple 2	0.0271	0.0827	-0.0744
Triple 3	-0.0798	0.2125	-0.0744
Triple 4	0.0142	-0.3965	0.0366
Triple 5	0.0160	0.2151	0.0205

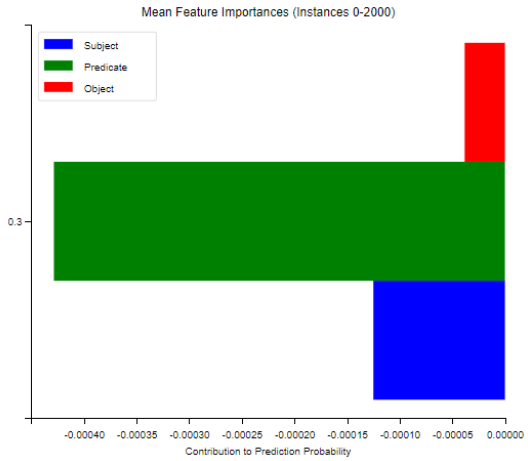
HoIE


Training vs Validation Loss
Training Loss
Validation Loss
Loss
Epoch

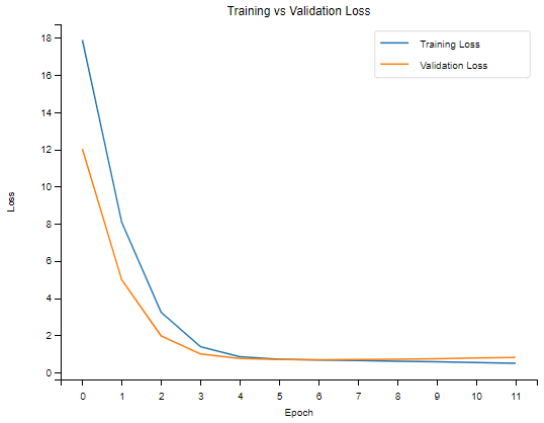
Precision-Recall Curve
Precision
Recall

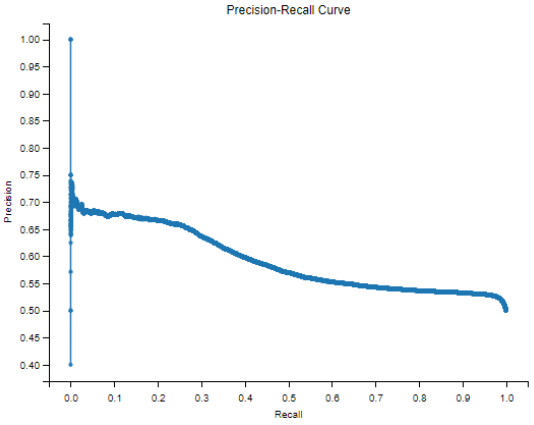

Calibration Curve
Model
Perfectly calibrated
Fraction of Positives
Mean Predicted Probability

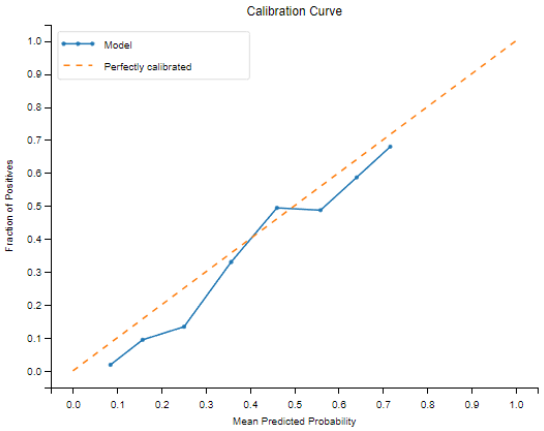

Confusion Matrix
16549
29572
11474
34638
True Labels
Predicted Labels

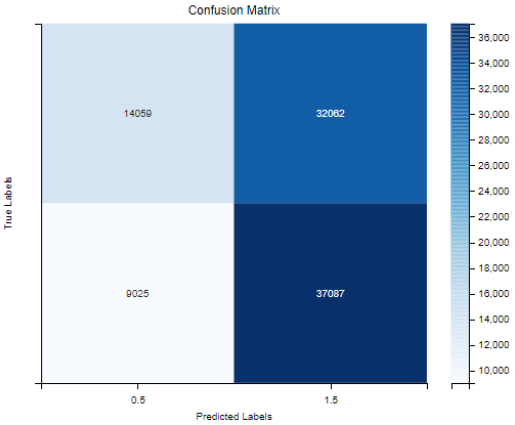

Mean Feature Importances (Instances 0-2000)
Subject
Predicate
Object
Contribution to Prediction Probability

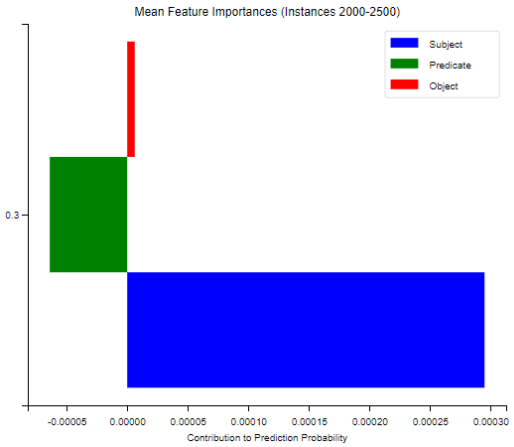
DistMult

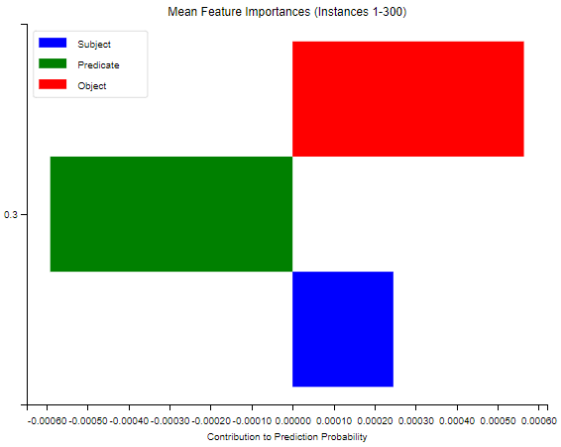


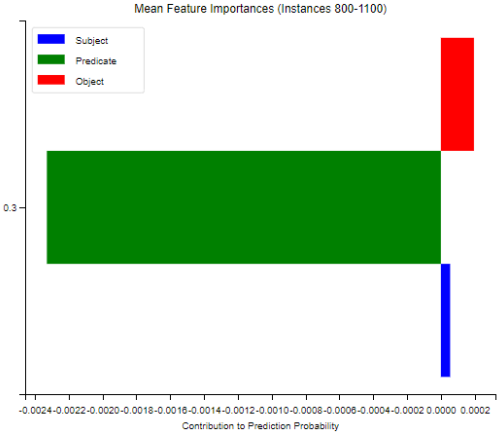






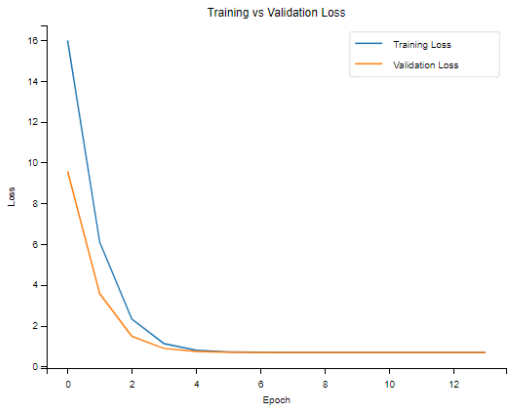


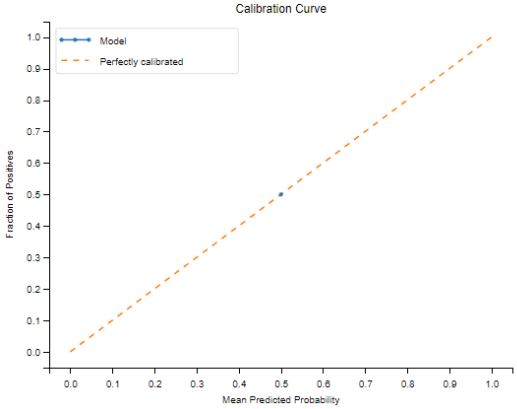




	subject	**predicate**	**object**
Triple 1	-0.0209	0.0337	0.0766
Triple 2	0.0450	-0.0647	0.0246
Triple 3	-0.0121	0.04366	-0.0549
Triple 4	-0.0130	-0.0628	-0.0326
Triple 5	-0.0083	0.0338	-0.0379

$ComplEx$





	C1	C2	C3	C4	C5	C6	C7	C8
$TransE$	1	0	0	0	0.5003	0.5003	0.5003	0
$TransH$	8	576	0	0	0.4927	0.8640	0.0001	0.1885
$RotatE$	4	34	0	0	0.3859	0.6597	0.0044	0.1988
$HolE$	39	33	0	0	0.4725	0.6735	0.0319	0.1037
$DistMult$	25	75	0	0	0.5022	0.6872	0.0480	0.0880
$ComplEx$	1	0	0	0	0.5003	0.5003	0.5003	0

$TransH$

Top 5 most frequent relations:
[('http://www.w3.org/1999/02/22-rdf-syntax-ns#rest', 816),
('http://www.w3.org/1999/02/22-rdf-syntax-ns#first', 620),
('http://www.w3.org/2002/07/owl#unionOf', 34),
('http://www.w3.org/2002/07/owl#versionInfo', 13),
('http://www.w3.org/2000/01/rdf-schema#subPropertyOf', 5)]
Relations appearing only once:
['https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasURL']
Relation with highest average probability:
('http://www.w3.org/2000/01/rdf-schema#subPropertyOf', 0.5982378005981446)
Relation with lowest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasOECDTheme',
0.44281700160354376)

$RotatE$

Top 5 most frequent relations:
[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/term', 1026),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/level', 307),

('http://www.w3.org/1999/02/22-rdf-syntax-ns#type', 142),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasReference', 25)
Relation with highest average probability:
('http://www.w3.org/1999/02/22-rdf-syntax-ns#type', 0.3978518114202249)
Relation with lowest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/level', 0.36762830644621475)

$HoIE$

Top 5 most frequent relations:
[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasCode', 429),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasURI', 251),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/term', 148),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/definition', 84),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/context', 57)]
Relations appearing only once:
['http://www.w3.org/2000/01/rdf-schema#domain',
'http://www.w3.org/2000/01/rdf-schema#subPropertyOf',
'http://www.w3.org/2002/07/owl#unionOf']
Relation with highest average probability:
('http://www.w3.org/2000/01/rdf-schema#subPropertyOf', 0.5536947846412659)
Relation with lowest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/fileLink', 0.38551637530326843)

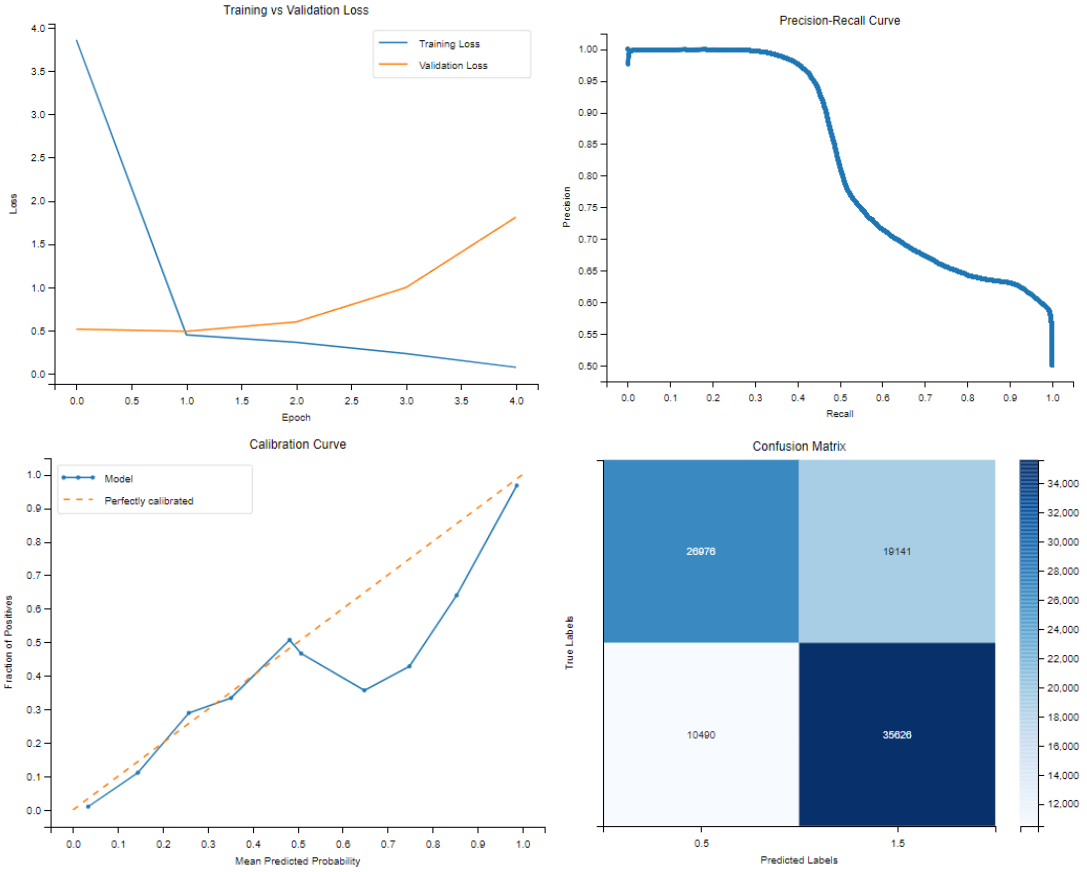
$DistMult$

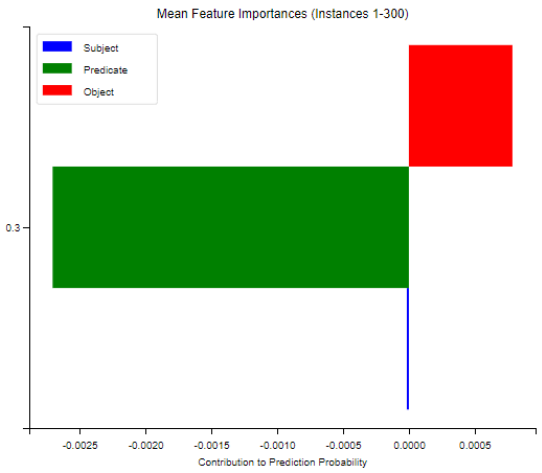
Top 5 most frequent relations:
[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasCode', 539),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/context', 264),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/title', 212),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/keyword', 86),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasURI', 66)]
Relations appearing only once:
['http://www.w3.org/2002/07/owl#imports', 'http://www.w3.org/2000/01/rdf-schema#label',
'https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasCategoryOfGlossaryArticle',
'http://www.w3.org/2000/01/rdf-schema#comment']
Relation with highest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasFrequentTerm',
0.5834585626920065)
Relation with lowest average probability:
('http://www.w3.org/2002/07/owl#imports', 0.39941537380218506)

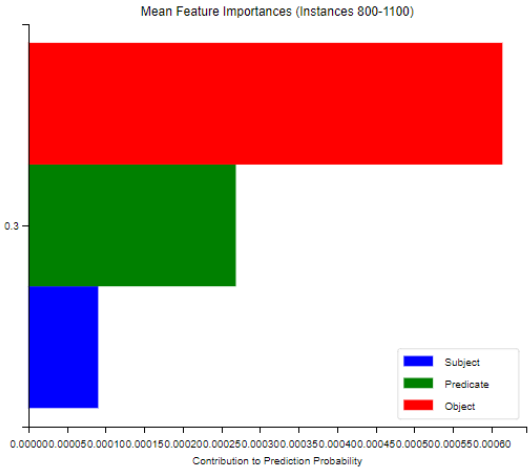
IV. Group 4

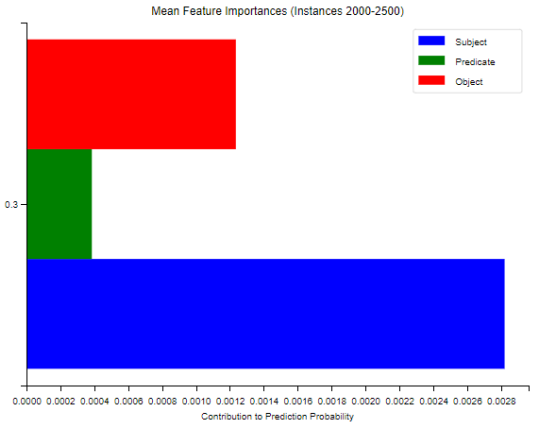
	Mean Silhouette	Mean Davies-Bouldin	Mean Calinski-Harabasz
$TransE$	0.8104	0.2439	3590.5302
$TransH$	0.8048	0.2519	3462.0251
$NoEmbds$	0.7977	0.2641	3273.4176
$HoIE$	0.7958	0.2646	3188.0821
$DistMult$	0.8024	0.2527	3642.3019
$ComplEx$	**0.8065**	**0.2480**	**3854.4008**

$TransE$



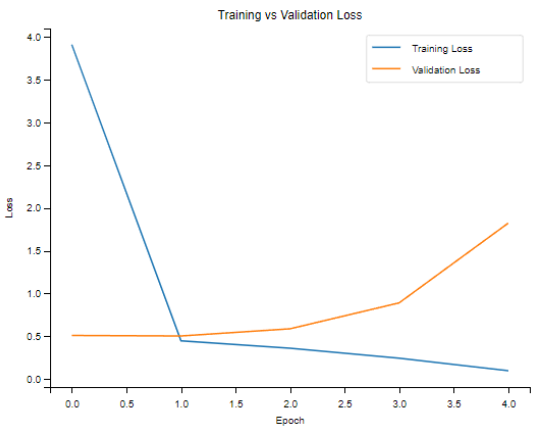


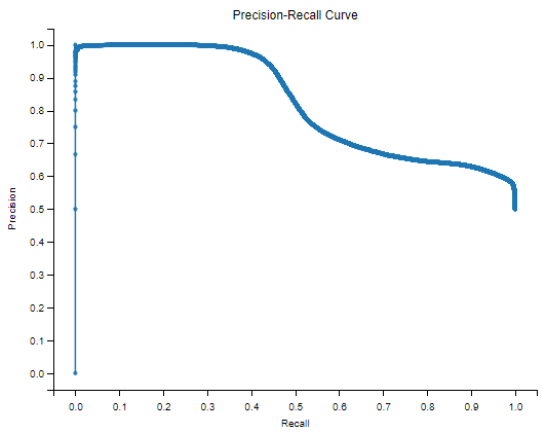


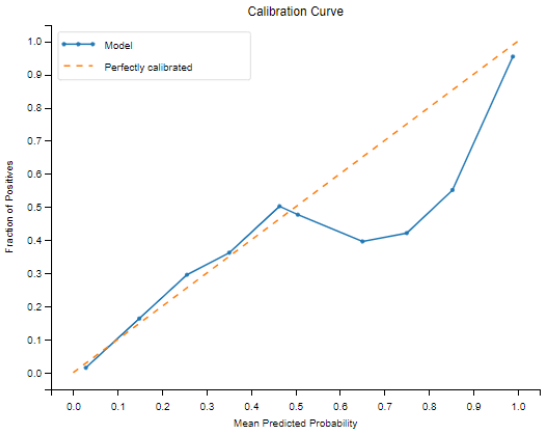


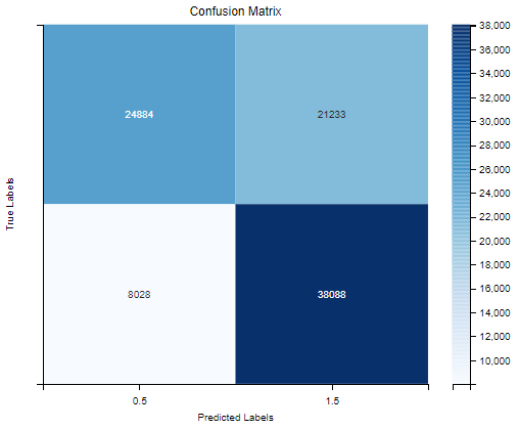
	subject	predicate	object
Triple 1	0.0007	0.03489	-0.0096
Triple 2	0.0008	0.03004	0.0264
Triple 3	-0.0042	0.03679	-0.0112
Triple 4	0.0019	-0.0630	-0.0022
Triple 5	-0.0044	0.01175	-0.0120

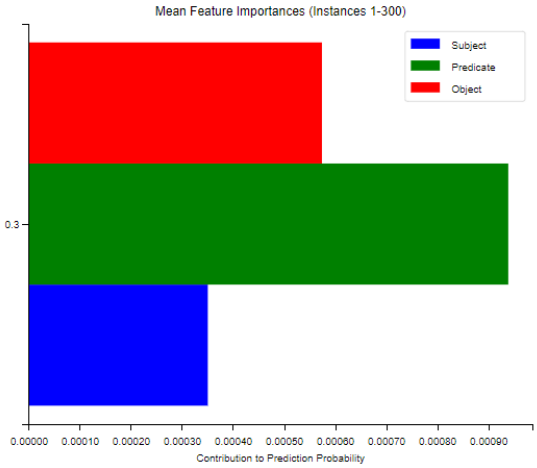
$TransH$

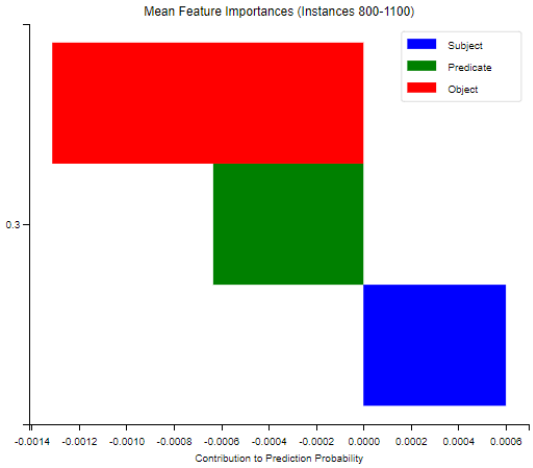


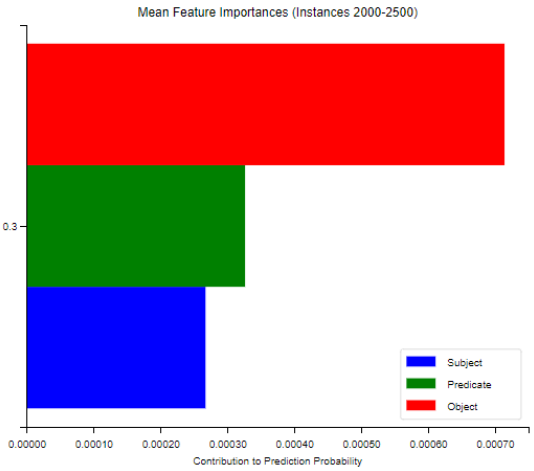






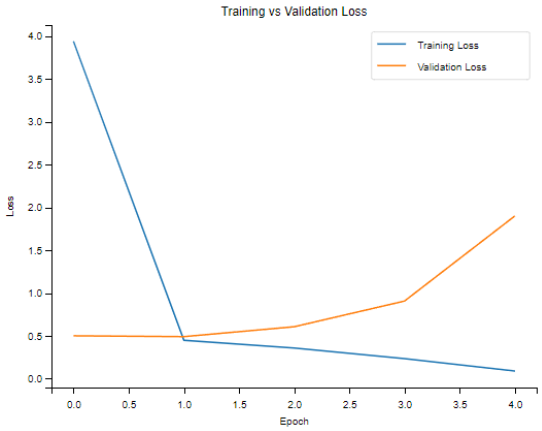


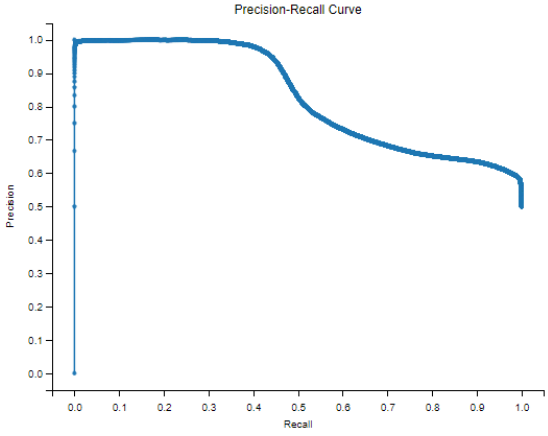


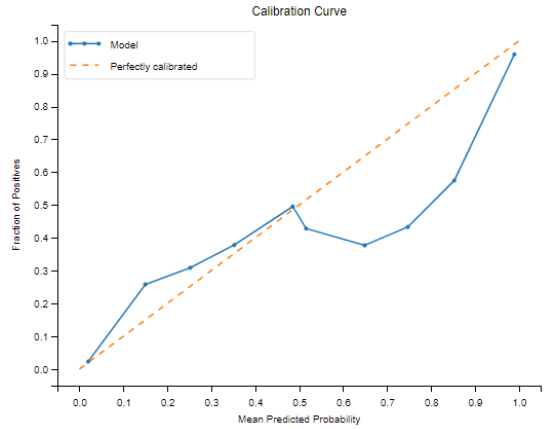


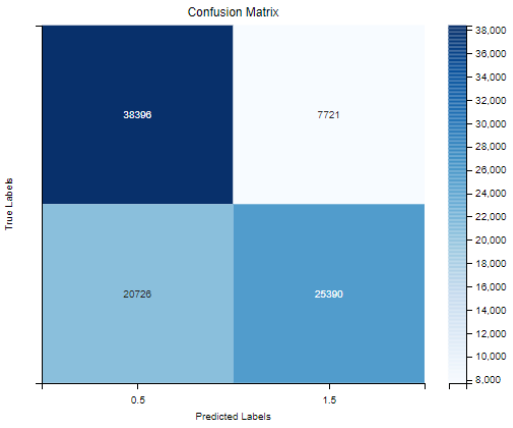
	subject	predicate	object
Triple 1	-0.0062	0.0970	0.2345
Triple 2	0.0099	0.3176	-0.0751
Triple 3	0.0033	-0.3573	-0.0222
Triple 4	-0.0039	-0.3620	-0.0815
Triple 5	0.0089	0.3167	-0.0562

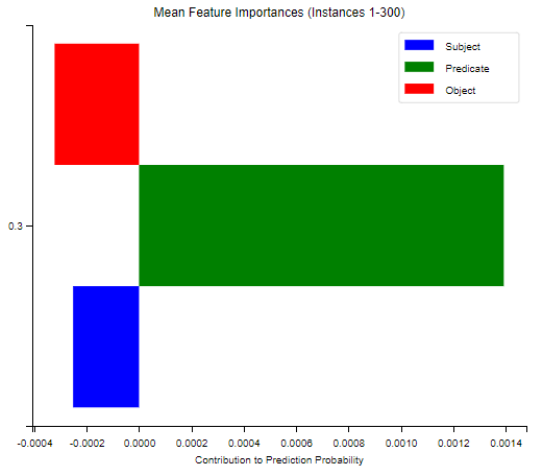
NoEmbds

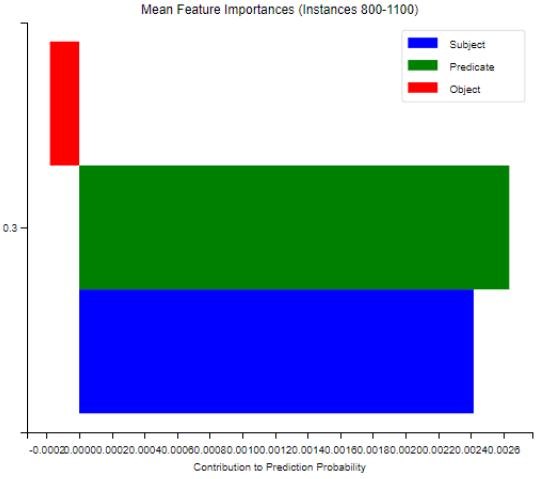

Training vs Validation Loss
Training Loss
Validation Loss
Loss
Epoch

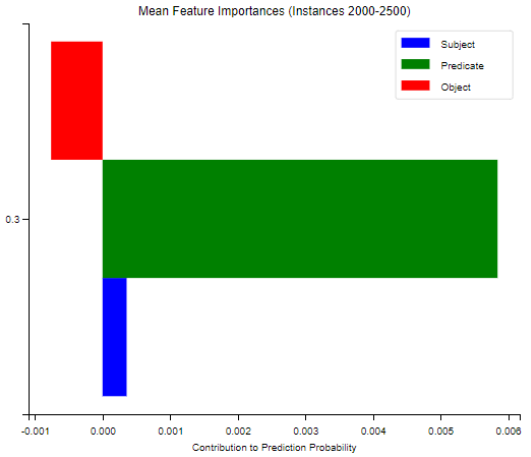

Precision-Recall Curve
Precision
Recall


Calibration Curve
Model
Perfectly calibrated
Fraction of Positives
Mean Predicted Probability


Confusion Matrix
38396
7721
20726
25390
True Labels
Predicted Labels

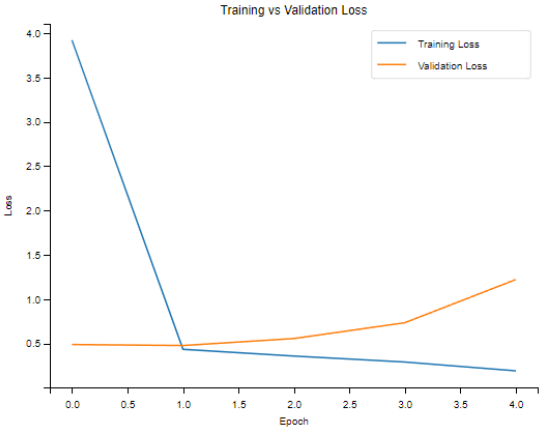

Mean Feature Importances (Instances 1-300)
Subject
Predicate
Object
Contribution to Prediction Probability

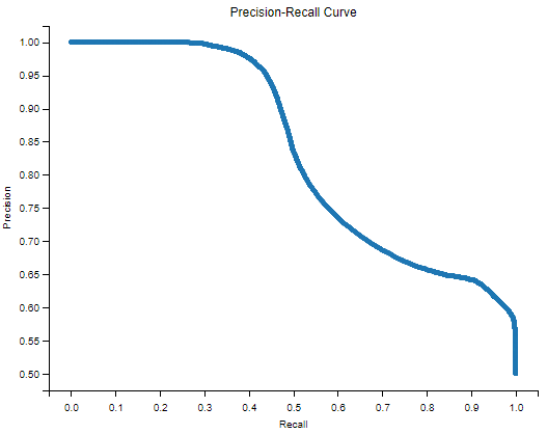

Mean Feature Importances (Instances 800-1100)
Subject
Predicate
Object
Contribution to Prediction Probability

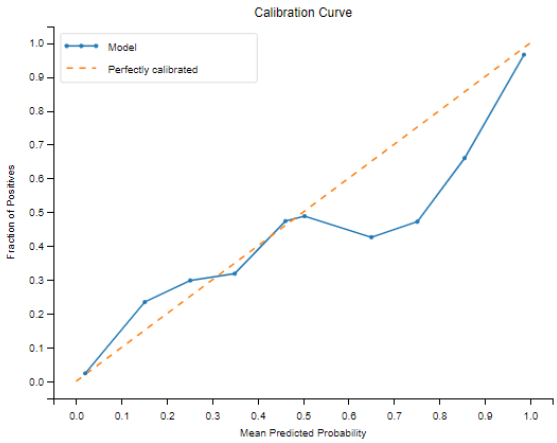


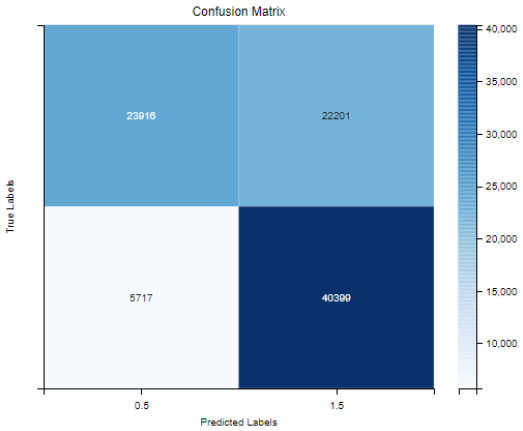
	subject	predicate	object
Triple 1	0.0147	0.1431	0.0033
Triple 2	0.0127	0.18140	-0.0192
Triple 3	-0.0166	0.14519	0.0757
Triple 4	-0.0180	-0.47417	-0.0395
Triple 5	0.0129	0.17572	0.0026

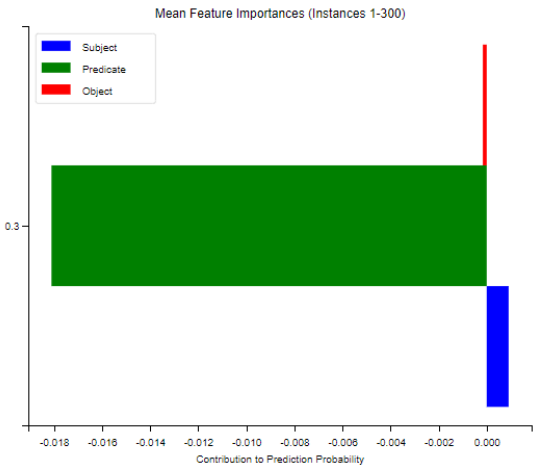
$HoIE$

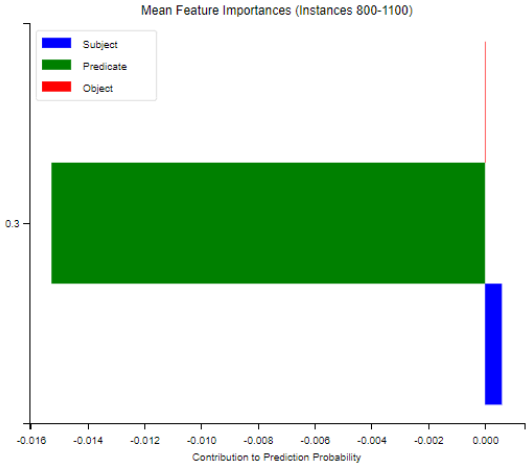


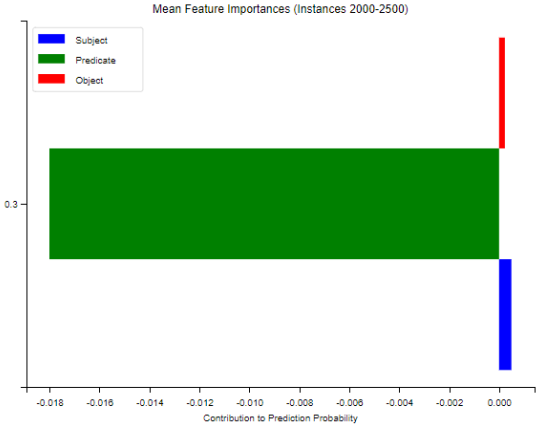






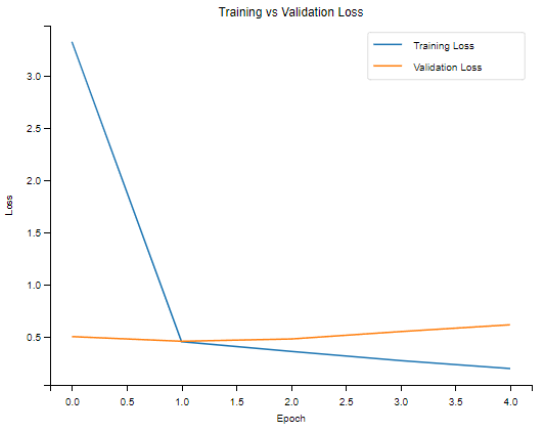


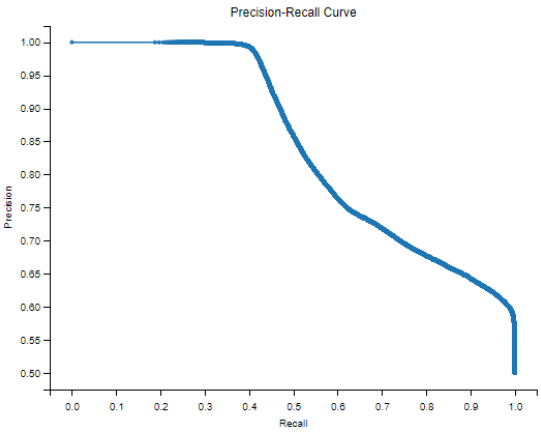


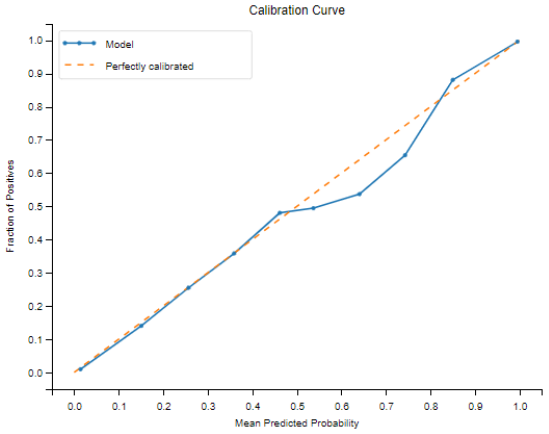


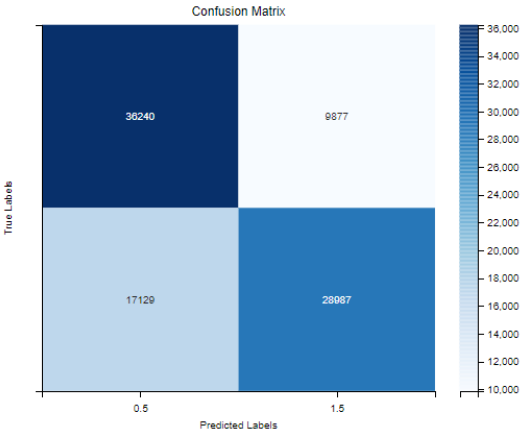
	subject	predicate	object
Triple 1	0.0030	-0.0297	-0.0335
Triple 2	4.1110	0.02921	0.0098
Triple 3	-0.0023	0.03722	-0.0264
Triple 4	-0.0044	-0.00908	0.0194
Triple 5	0.0023	-0.0121	0.0145

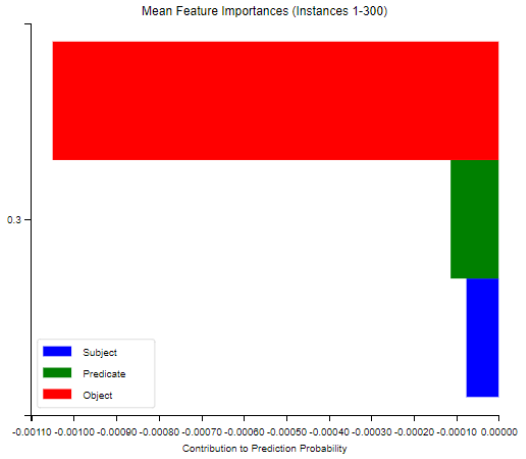
DistMult

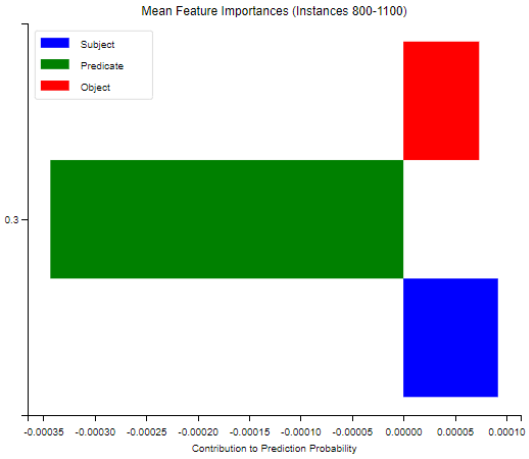


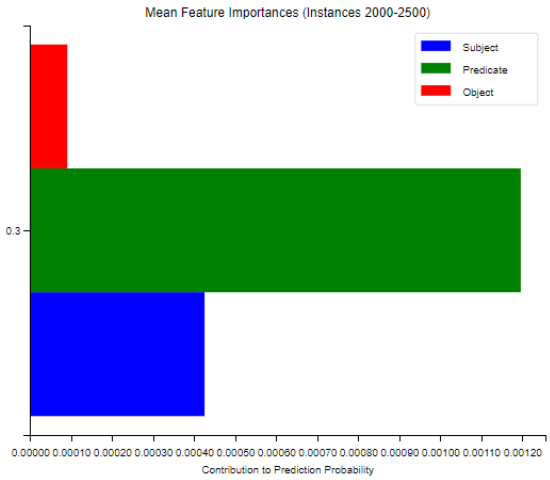






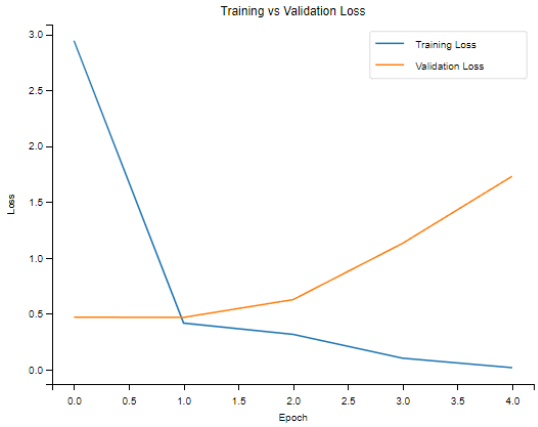


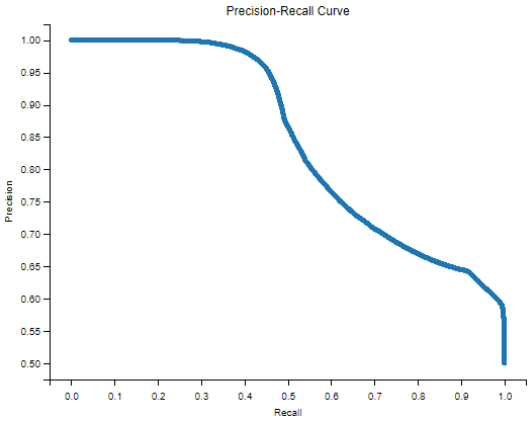


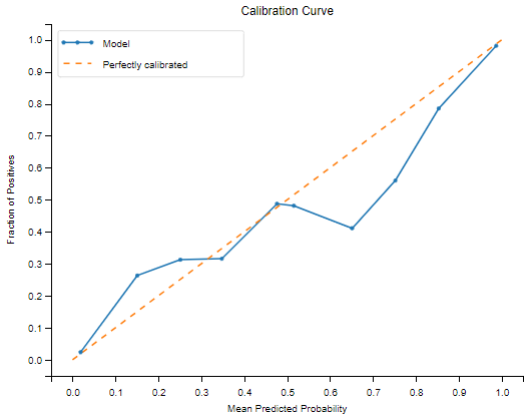


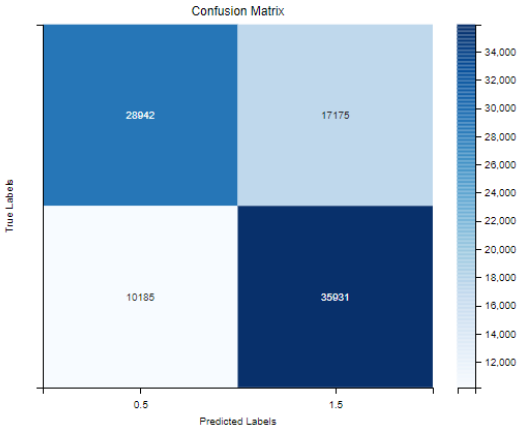
	subject	predicate	object
Triple 1	0.0057	0.10279	-0.0066
Triple 2	0.0006	0.0055	0.0313
Triple 3	-0.0013	0.0200	-0.0211
Triple 4	-0.0004	-0.0603	-0.0063
Triple 5	0.0026	-0.0338	0.0472

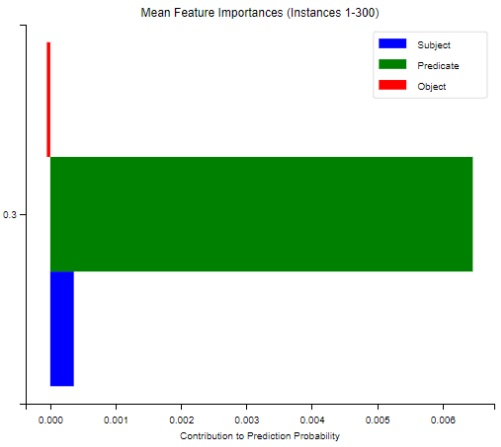
ComplEx

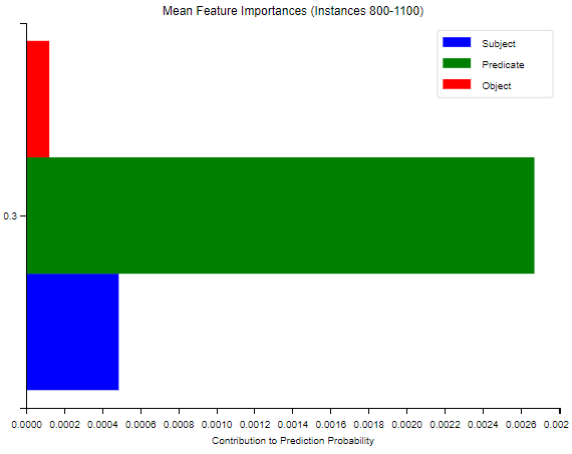

Training vs Validation Loss
Training Loss
Validation Loss
Loss
Epoch

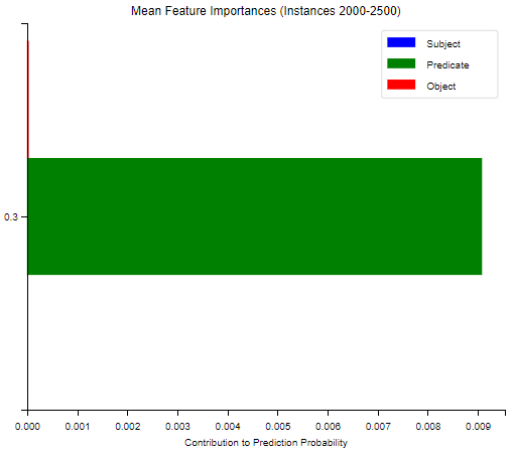

Precision-Recall Curve
Precision
Recall


Calibration Curve
Model
Perfectly calibrated
Fraction of Positives
Mean Predicted Probability


Confusion Matrix
28942
17175
10185
35931
True Labels
Predicted Labels


Mean Feature Importances (Instances 1-300)
Subject
Predicate
Object
Contribution to Prediction Probability


Mean Feature Importances (Instances 800-1100)
Subject
Predicate
Object
Contribution to Prediction Probability



	subject	predicate	object
Triple 1	-0.0179	-0.1792	-0.1587
Triple 2	0.0373	0.1316	0.0274
Triple 3	-0.0148	-0.1770	-0.1598
Triple 4	-0.0262	-0.1628	0.0199
Triple 5	0.0031	0.0999	0.1665

	C1	C2	C3	C4	C5	C6	C7	C8
$TransE$	3	133	15	3	0.3830	0.9862	0.0002	0.2360
$TransH$	3	193	50	2	0.3646	0.9998	0.0009	0.2445
$RotatE$	3	90	17	3	0.3648	0.9953	0.0001	0.2239
$HolE$	4	94	20	4	0.4152	0.9881	0.0007	0.1975
$DistMult$	4	118	2	1	0.4340	0.9992	0.0174	0.1414
$ComplEx$	4	59	2	2	0.4693	0.9691	0.0272	0.1148

$TransE$

Top 5 most frequent relations:
[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/term', 822),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasReference', 612),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/level', 66)]
Relation with highest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/level', 0.42565073893373745)
Relation with lowest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasReference',
0.37573392293734587)

$TransH$

Top 5 most frequent relations:
[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/term', 955),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/level', 519),
('http://www.w3.org/1999/02/22-rdf-syntax-ns#type', 26)]
Relations appearing only once:
Relation with highest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/term', 0.3720383586696013)
Relation with lowest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/level', 0.35139716155167583)

$NoEmbds$

Top 5 most frequent relations:
[('http://www.w3.org/1999/02/22-rdf-syntax-ns#type', 662),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/term', 527),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/level', 207)]]
Relation with highest average probability:
('http://www.w3.org/1999/02/22-rdf-syntax-ns#type', 0.3745577901346935)
Relation with lowest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/term', 0.35538292570659313)

$HoIE$

Top 5 most frequent relations:
[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasReference', 1216),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/level', 184),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/term', 99),
('http://www.w3.org/1999/02/22-rdf-syntax-ns#type', 1)]
Relations appearing only once:
['http://www.w3.org/1999/02/22-rdf-syntax-ns#type']
Relation with highest average probability:
('http://www.w3.org/1999/02/22-rdf-syntax-ns#type', 0.9866275191307068)
Relation with lowest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasReference',
0.4092006692742296)

$DistMult$

Top 5 most frequent relations:
[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/level', 630),
('http://www.w3.org/1999/02/22-rdf-syntax-ns#type', 536),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/term', 237),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasReference', 97)]
Relation with highest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/level', 0.44132194666280633)
Relation with lowest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/term', 0.4271914344516736)

$ComplEx$

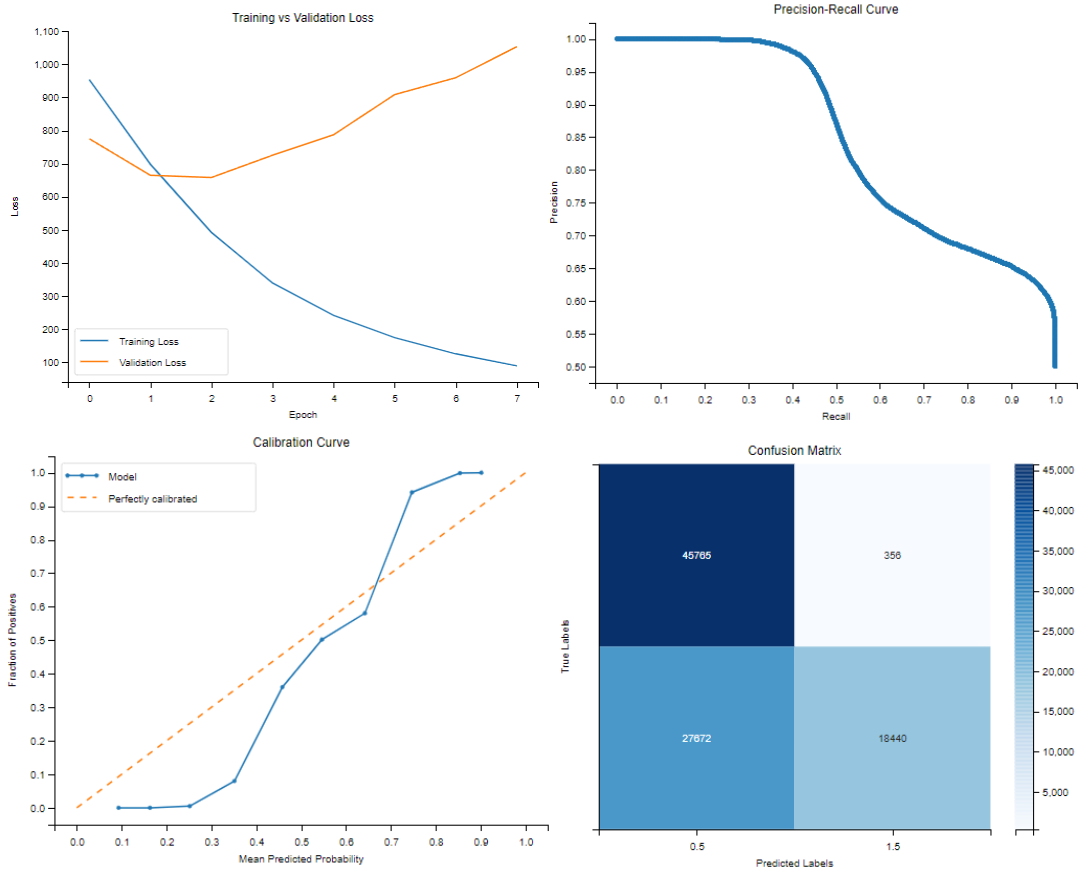
Top 5 most frequent relations:
[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/term', 574),
('http://www.w3.org/1999/02/22-rdf-syntax-ns#type', 446),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/level', 328),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasReference', 152)]
Relation with highest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/level', 0.48241893212288256)

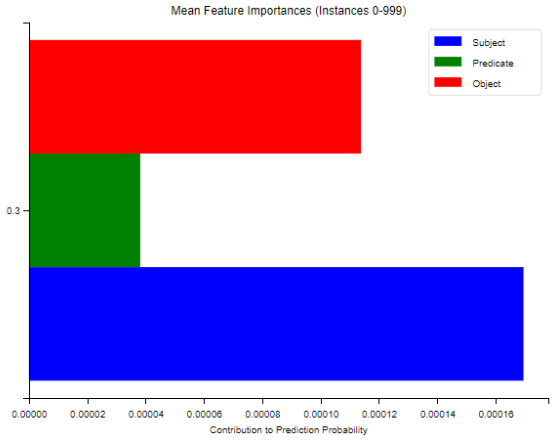
Relation with lowest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasReference',
0.4608497160339826)

V. Group 5

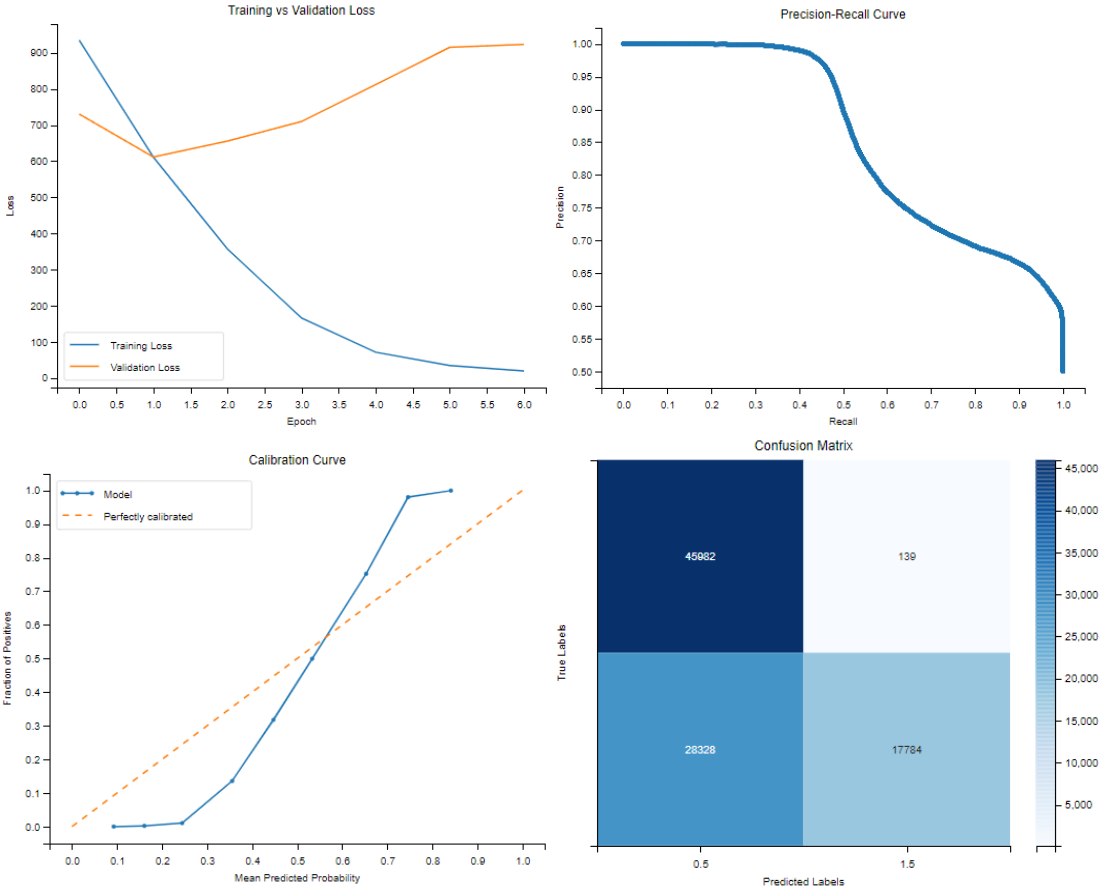
	Mean Silhouette	Mean Davies-Bouldin	Mean Calinski-Harabasz
$TransE$	0.8121	0.2407	4119.5141
$TransH$	0.8130	0.2393	4052.4581
$RotatE$	0.7870	0.2817	3135.0636
$HoIE$	0.8064	0.2497	3473.7874
$DistMult$	**0.8120**	**0.2396**	**4655.1074**
$ComplEx$	0.8031	0.2532	3555.0818

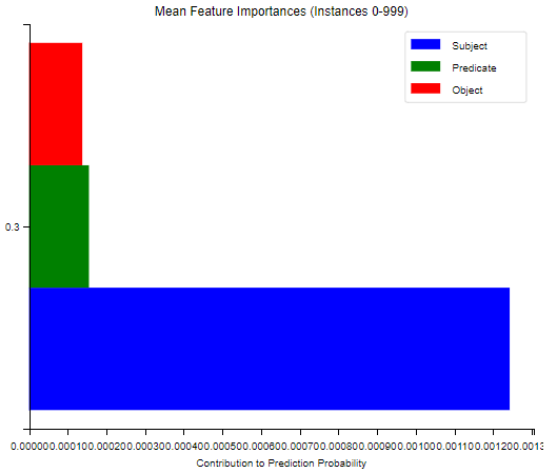
$TransE$



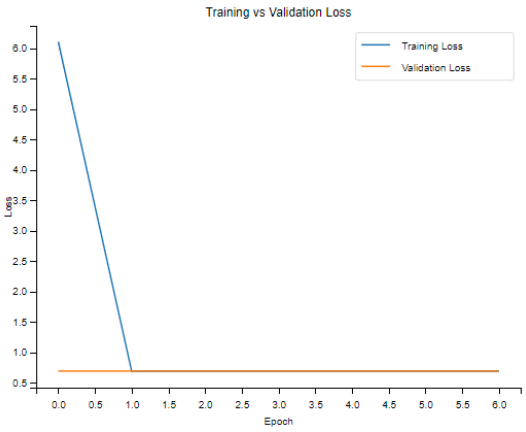

Mean Feature Importances (Instances 0-999)
Subject
Predicate
Object
0.3
0.00000
0.00002
0.00004
0.00006
0.00008
0.00010
0.00012
0.00014
0.00016
Contribution to Prediction Probability

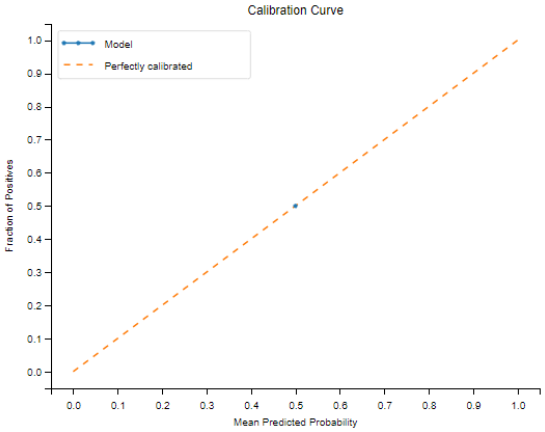
$TransH$


Training vs Validation Loss
Loss
Epoch
Training Loss
Validation Loss
Precision-Recall Curve
Precision
Recall
Calibration Curve
Model
Perfectly calibrated
Fraction of Positives
Mean Predicted Probability
Confusion Matrix
45982
139
28328
17784
True Labels
Predicted Labels

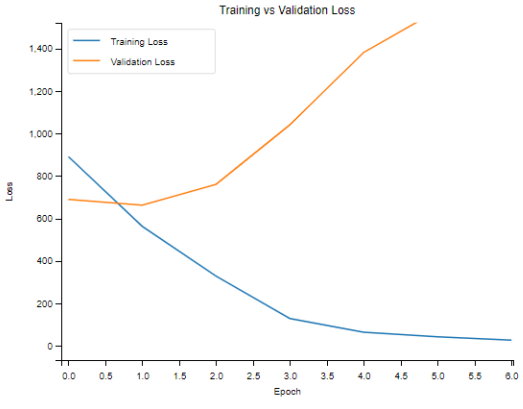

Mean Feature Importances (Instances 0-999)
Subject
Predicate
Object
0.3
Contribution to Prediction Probability

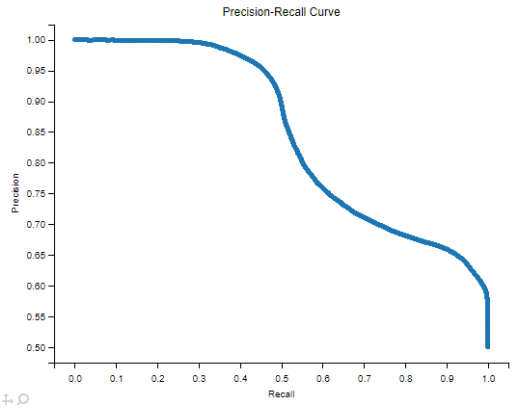
RotatE

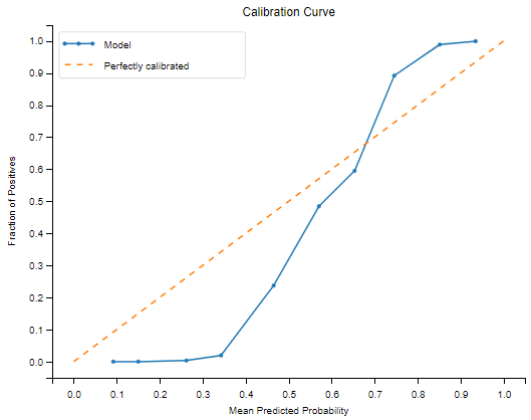

Training vs Validation Loss
Training Loss
Validation Loss
Loss
Epoch

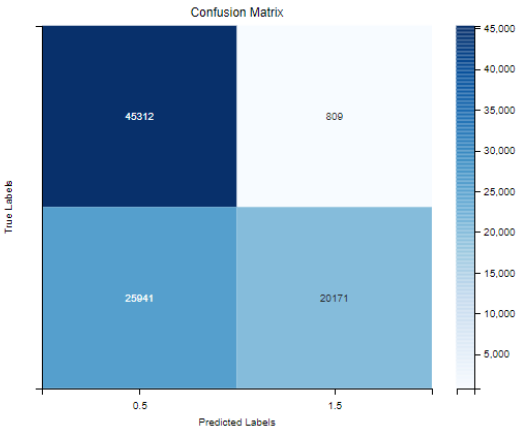

Calibration Curve
Model
Perfectly calibrated
Fraction of Positives
Mean Predicted Probability

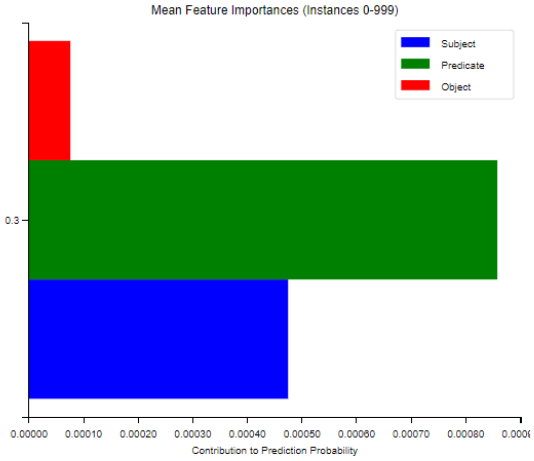
HoIE


Training vs Validation Loss
Training Loss
Validation Loss
Loss
Epoch


Precision-Recall Curve
Precision
Recall

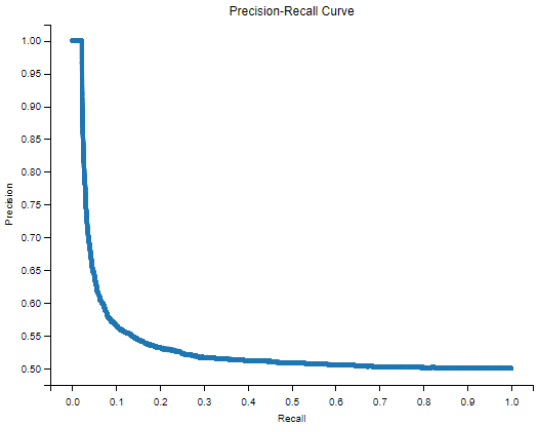

Calibration Curve
Model
Perfectly calibrated
Fraction of Positives
Mean Predicted Probability

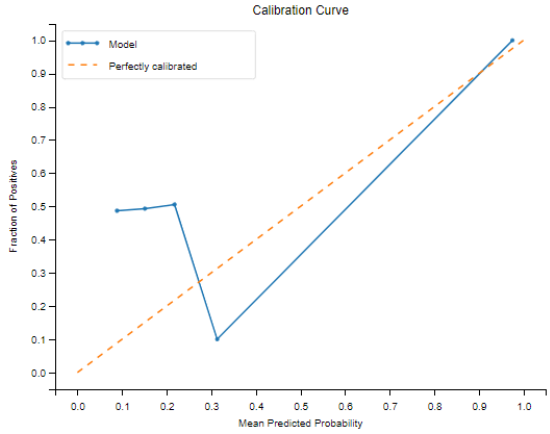

Confusion Matrix
45312
809
25941
20171
True Labels
Predicted Labels

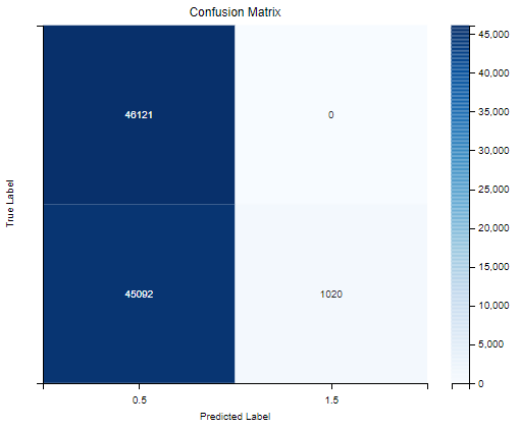

Mean Feature Importances (Instances 0-999)
Subject
Predicate
Object
Contribution to Prediction Probability

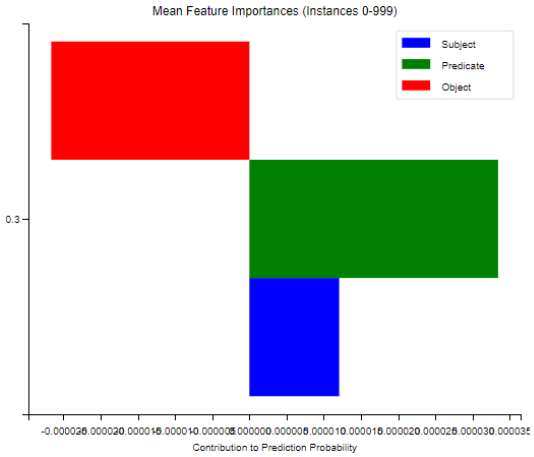
DistMult


Training vs Validation Loss
Training Loss
Validation Loss
Loss
Epoch

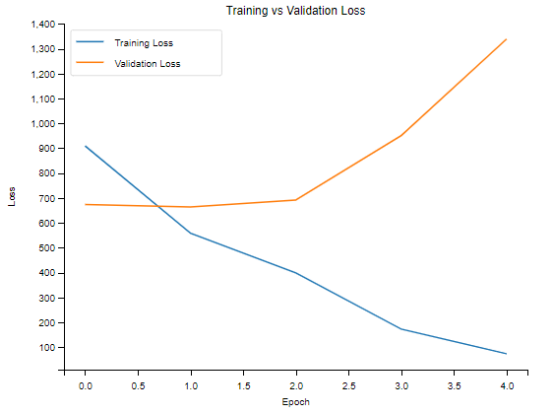

Precision-Recall Curve
Precision
Recall

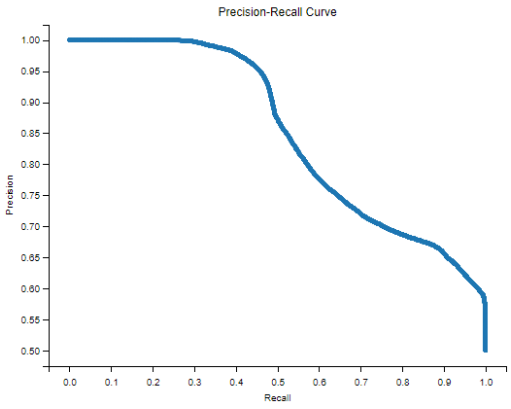

Calibration Curve
Model
Perfectly calibrated
Fraction of Positives
Mean Predicted Probability

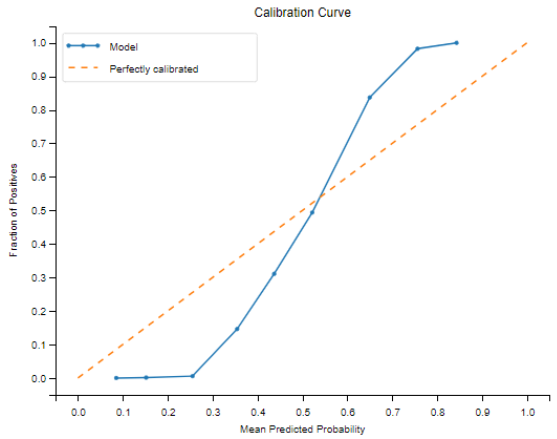

Confusion Matrix
46121
0
45002
1020
True Label
Predicted Label

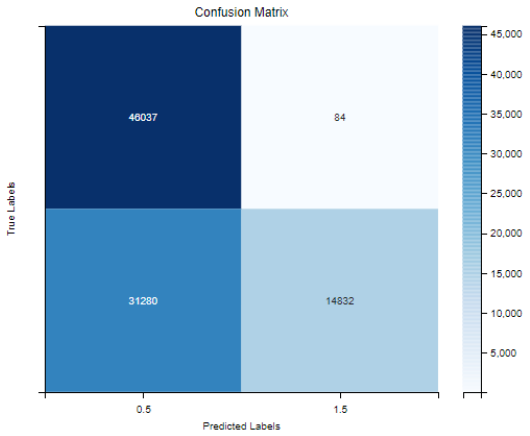

Mean Feature Importances (Instances 0-999)
Subject
Predicate
Object
Contribution to Prediction Probability

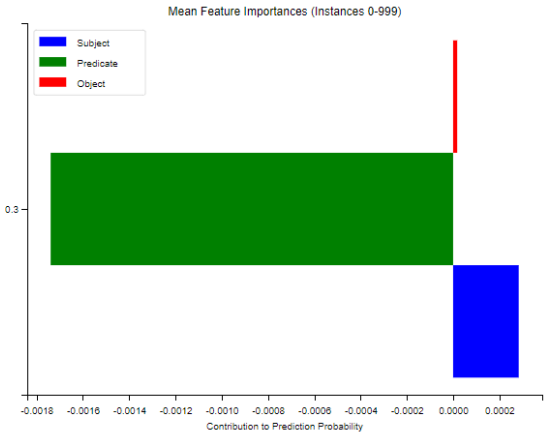
ComplEx


Training vs Validation Loss
Training Loss
Validation Loss
Loss
Epoch


Precision-Recall Curve
Precision
Recall







	C1	C2	C3	C4	C5	C6	C7	C8
$TransE$	10	140	0	0	0.4638	0.7433	0.0921	0.1239
$TransH$	17	51	0	0	0.4401	0.7201	0.0924	0.1235
$RotatE$	1	0	0	0	0.5000	0.5	0.5	0
$HoIE$	8	54	0	0	0.4827	0.7362	0.0931	0.1251
$DistMult$	43	0	0	0	0.1553	0.2747	0.0383	0.0278
$ComplEx$	8	53	0	0	0.4296	0.6770	0.0585	0.1212

$TransE$

Top 5 most frequent relations:

[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/id', 538),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasURI', 459),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasReference', 226),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/definition', 148),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasParagraph', 83)]

Relations appearing only once:

['https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasFrequentTerm']

Relation with highest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasFrequentTerm', 0.5402535200119019)
Relation with lowest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/content', 0.44181166047399695)

$TransH$

Top 5 most frequent relations:
[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/databasePath', 616),
('http://www.w3.org/2000/01/rdf-schema#subClassOf', 138),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasURI', 134),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/content', 117),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/term', 117)]
Relations appearing only once:
['https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasParagraph',
'https://ec.europa.eu/eurostat/NLP4StatRef/ontology/id',
'http://www.w3.org/2000/01/rdf-schema#label',
'https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasCode']
Relation with highest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasParagraph', 0.6234550476074219)
Relation with lowest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasCode', 0.2667442560195923)

$HoIE$

Top 5 most frequent relations:
[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasParagraph', 987),
('http://www.w3.org/2000/01/rdf-schema#subClassOf', 307),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasCode', 128),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/relatedTerm', 44),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasURL', 20)]
Relations appearing only once:
['https://ec.europa.eu/eurostat/NLP4StatRef/ontology/fileLink',
'https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasFrequentTerm']
Relation with highest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasFrequentTerm', 0.627998411655426)
Relation with lowest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/fileLink', 0.2694052457809448)

$DistMult$

Top 5 most frequent relations:
[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasCode', 346),
('http://www.w3.org/1999/02/22-rdf-syntax-ns#type', 94),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasURL', 88),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/term', 82),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/title', 76)]
Relations appearing only once:
['http://www.w3.org/2002/07/owl#backwardCompatibleWith']
Relation with highest average probability:
('http://www.w3.org/2002/07/owl#equivalentProperty', 0.17751998826861382)
Relation with lowest average probability:
('http://www.w3.org/2000/01/rdf-schema#range', 0.13417193790276846)

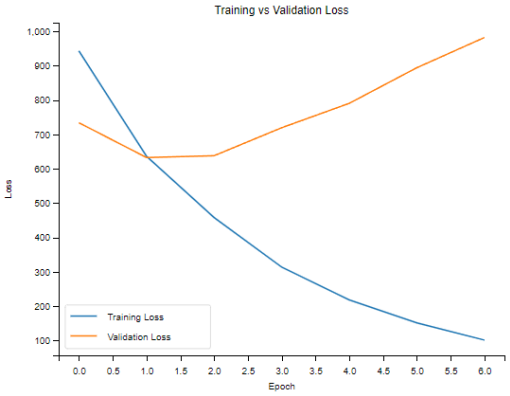
$ComplEx$

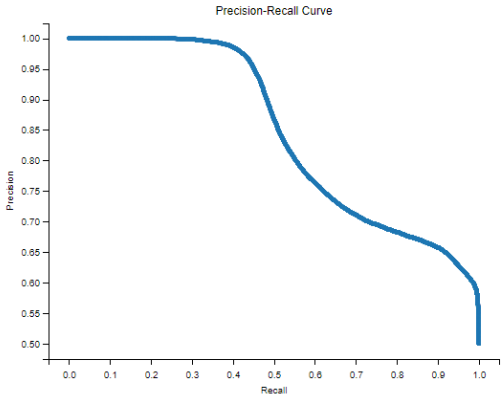
Top 5 most frequent relations:
[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/content', 707),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasURI', 368),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/term', 303),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasURL', 79),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasCode', 30)]
Relation with highest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/relatedTerm', 0.5032630264759064)
Relation with lowest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasURI', 0.41591848321663943)

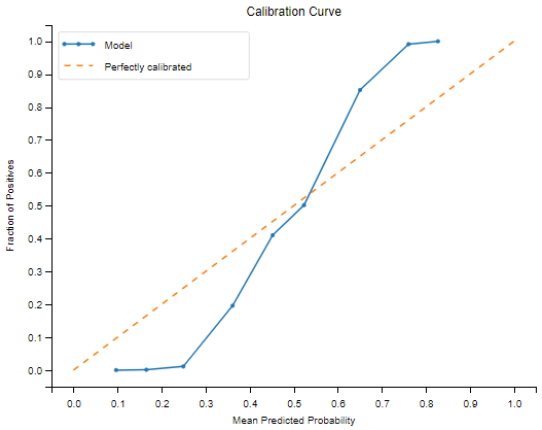
VI. Group 6

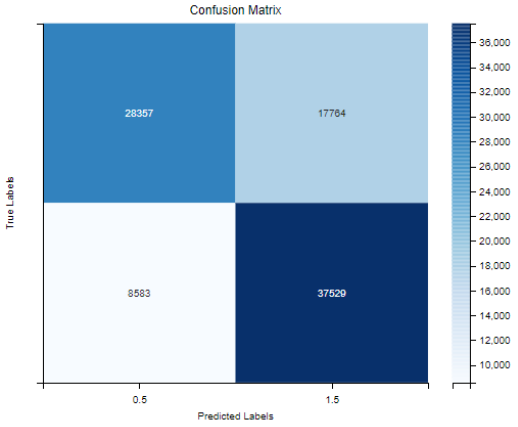
	Mean Silhouette	**Mean Davies-Bouldin**	**Mean Calinski-Harabasz**
$TransE$	0.8058	0.2516	3563.2767
$TransH$	0.8129	0.2398	3744.4443
$RotatE$	0.7941	0.2700	4132.4151
$HoIE$	**0.8129**	**0.2402**	**4226.9172**
$DistMult$	0.8186	0.2322	3862.0030
$ComplEx$	0.7928	0.2710	4206.7353

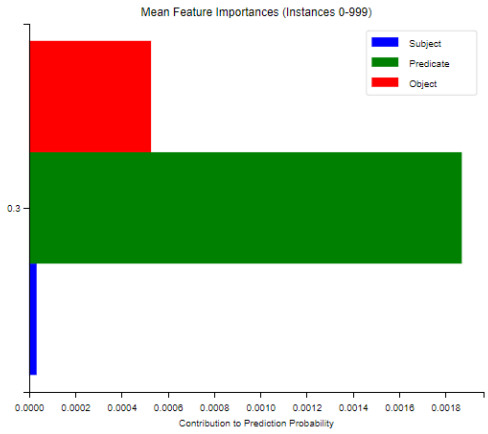
TransE


Training vs Validation Loss
Loss
Epoch
Training Loss
Validation Loss

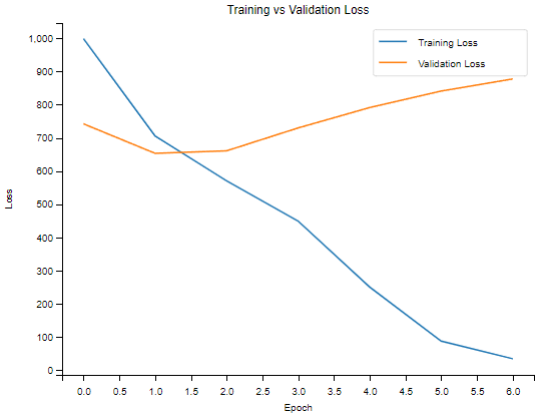

Precision-Recall Curve
Precision
Recall

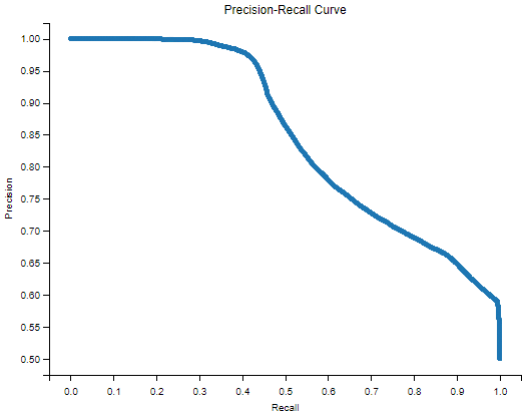

Calibration Curve
Model
Perfectly calibrated
Fraction of Positives
Mean Predicted Probability

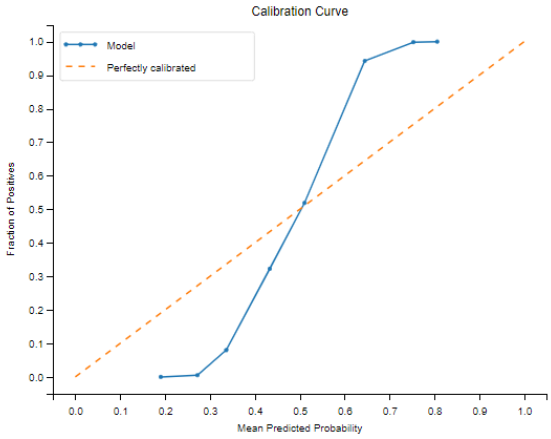

Confusion Matrix
28357
17764
8583
37529
True Labels
Predicted Labels

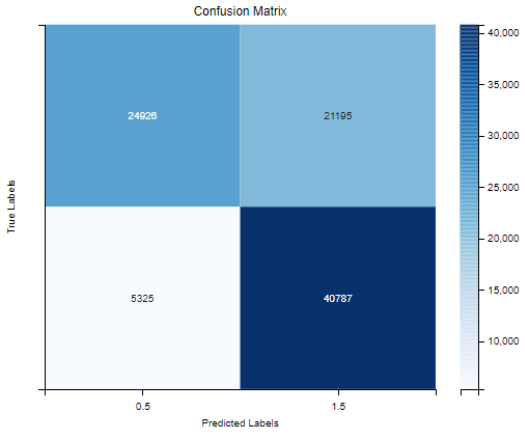

Mean Feature Importances (Instances 0-999)
Subject
Predicate
Object
Contribution to Prediction Probability

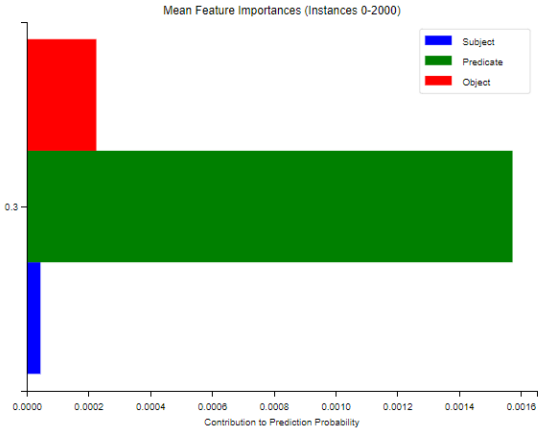
$TransH$


Training vs Validation Loss
Training Loss
Validation Loss
Loss
Epoch

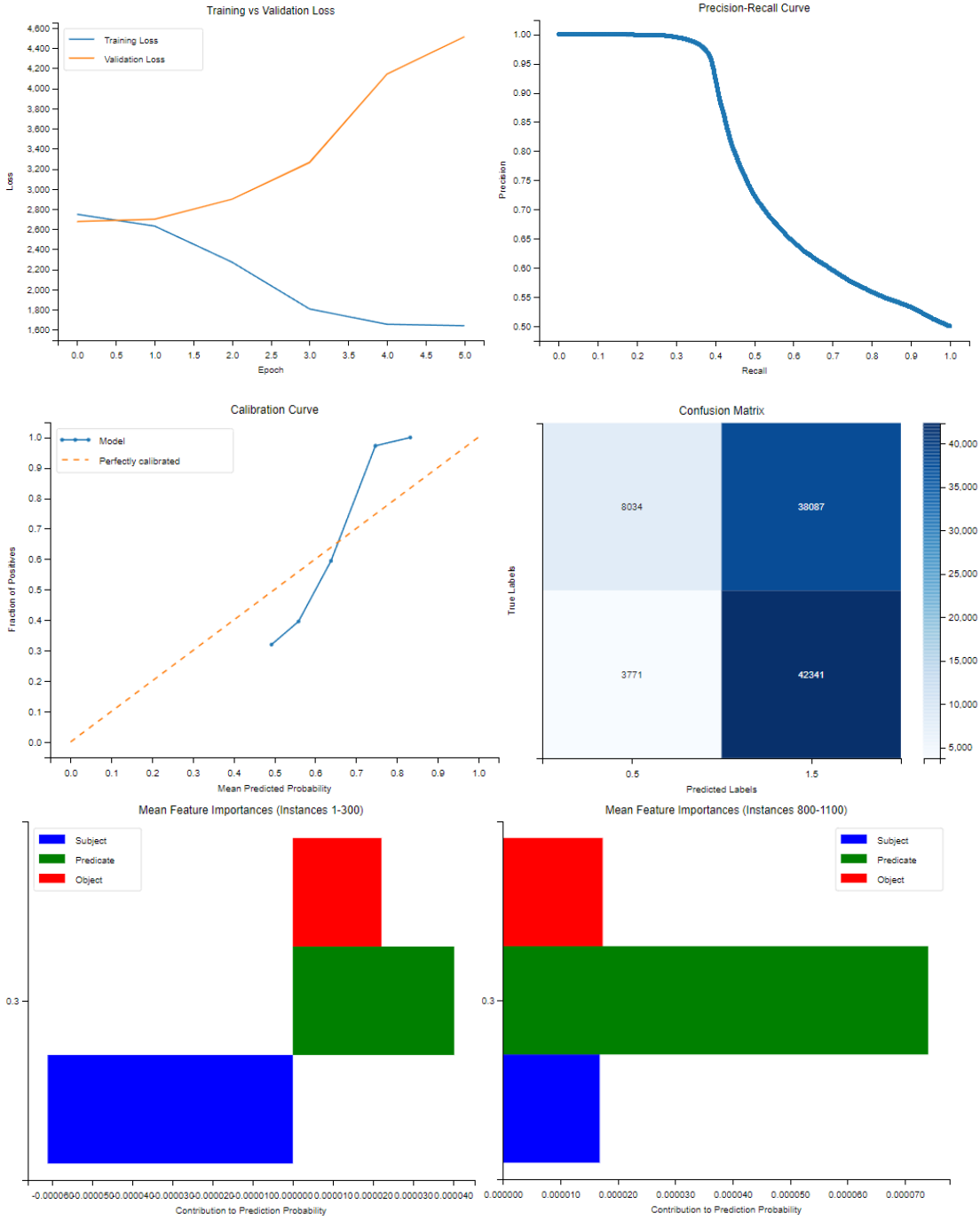

Precision-Recall Curve
Precision
Recall


Calibration Curve
Model
Perfectly calibrated
Fraction of Positives
Mean Predicted Probability

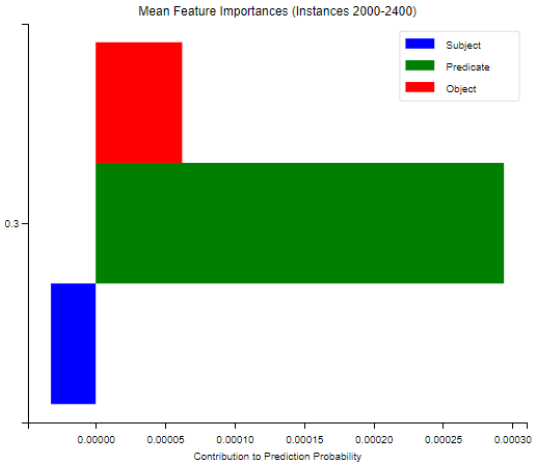

Confusion Matrix
24026
21195
5325
40787
True Labels
Predicted Labels


Mean Feature Importances (Instances 0-2000)
Subject
Predicate
Object
Contribution to Prediction Probability

$RotatE$

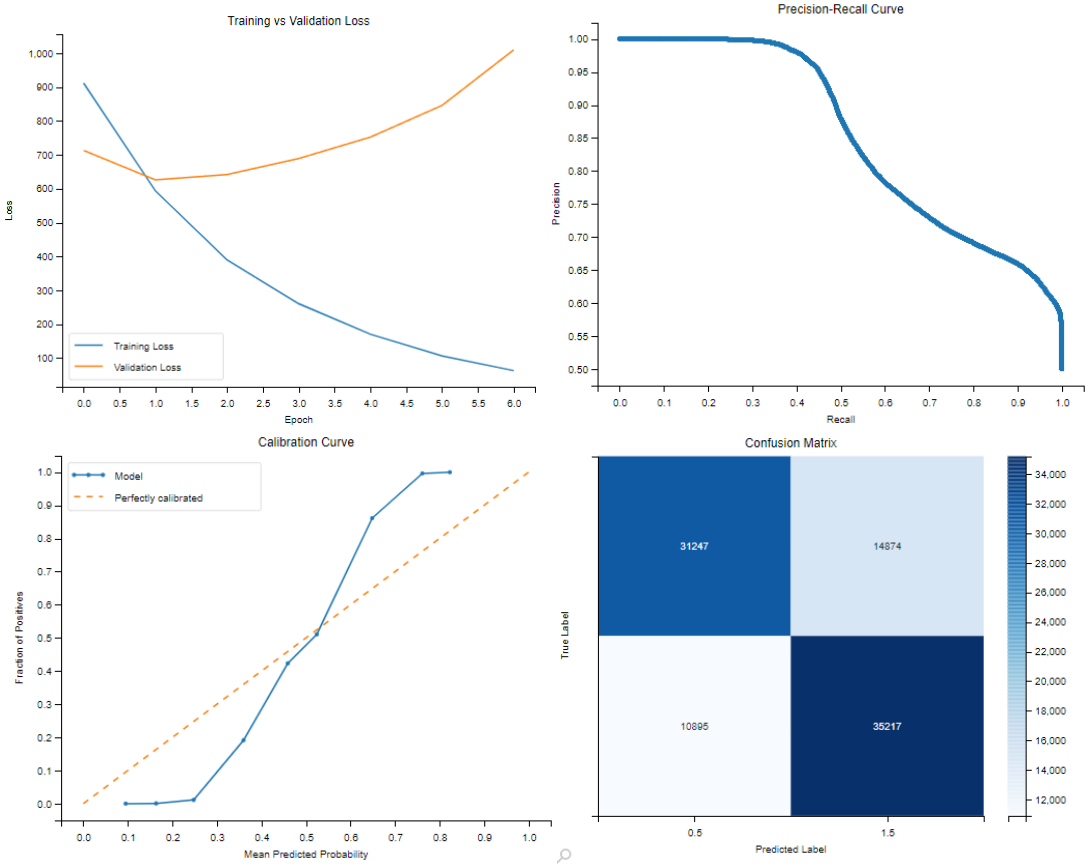


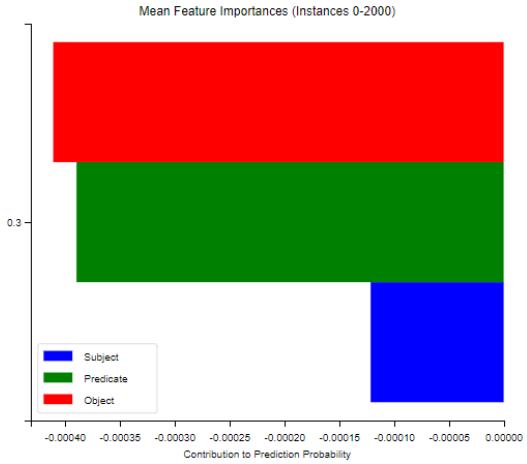
subject predicate object



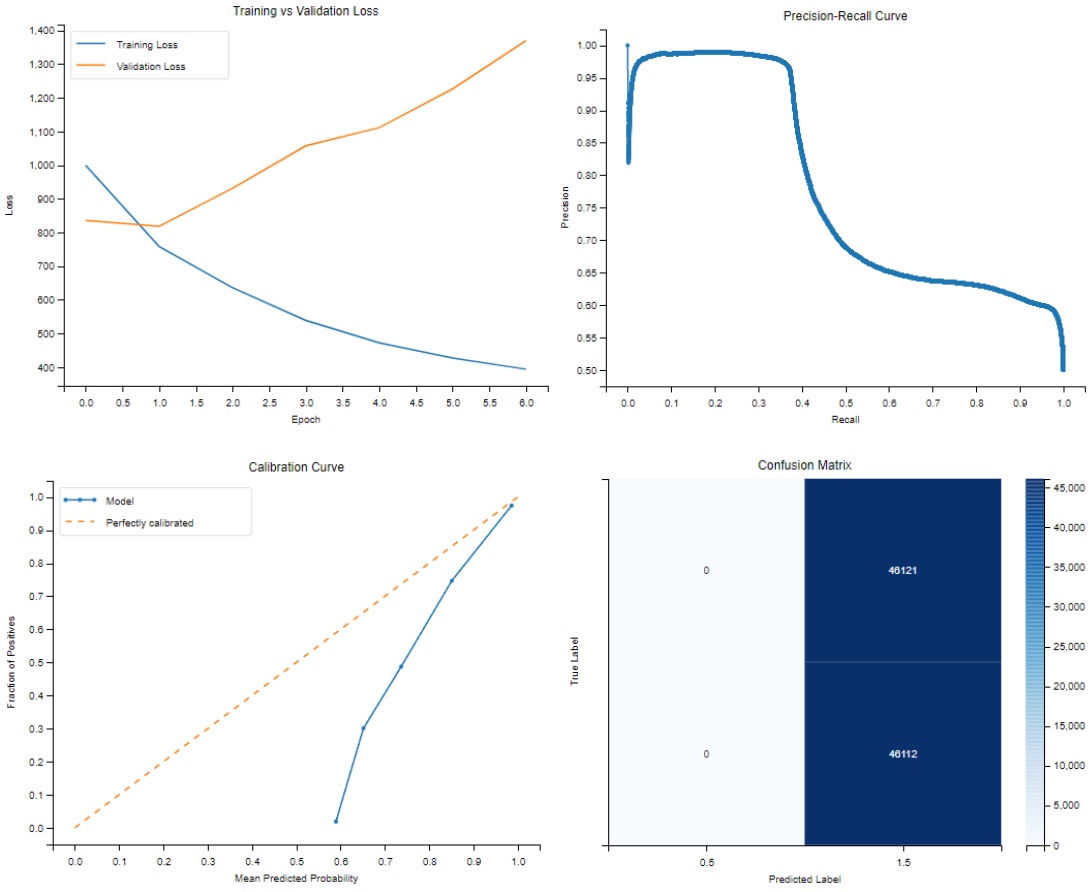
Triple 1	0.0006	-0.00036	0.0377
Triple 2	-0.0018	0.00010	0.0097
Triple 3	0.0033	-0.00074	0.0106
Triple 4	0.0008	-0.00051	0.0358
Triple 5	-0.0020	-0.00039	-0.0897

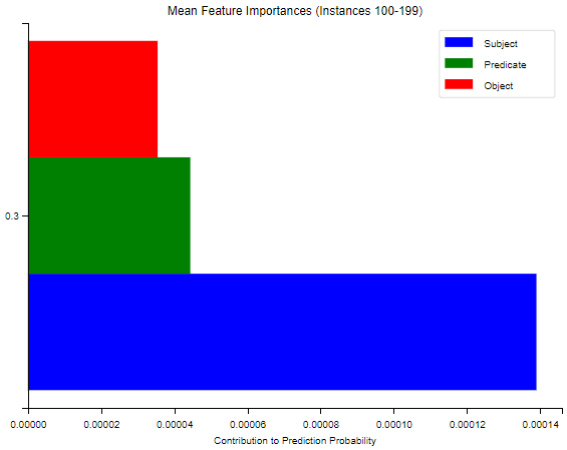
HoIE



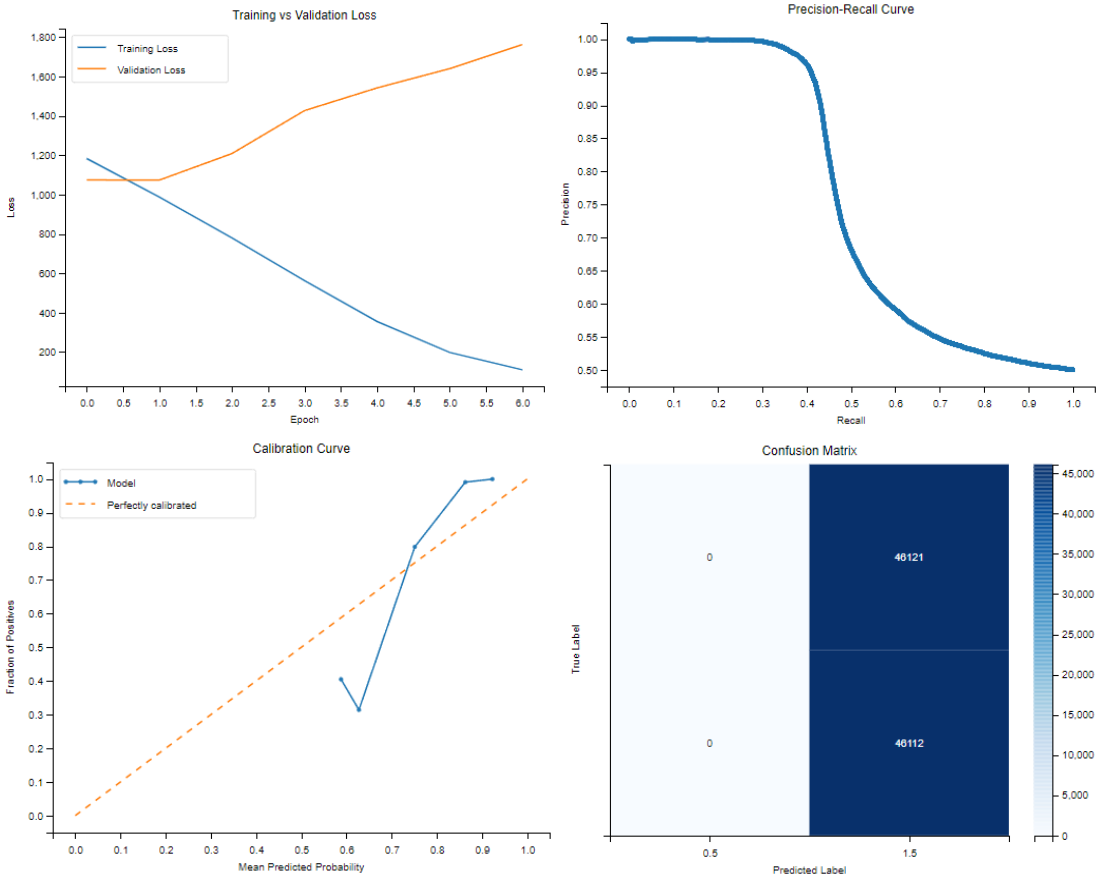

Mean Feature Importances (Instances 0-2000)
0.3
Subject
Predicate
Object
-0.00040
-0.00035
-0.00030
-0.00025
-0.00020
-0.00015
-0.00010
-0.00005
0.00000
Contribution to Prediction Probability

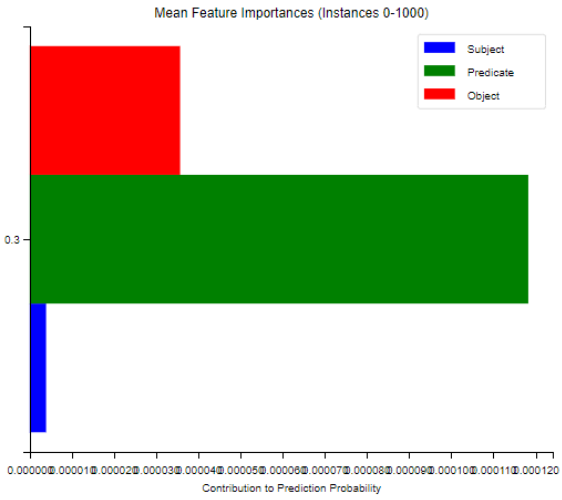
DistMult


Training vs Validation Loss
Training Loss
Validation Loss
Loss
Epoch
Precision-Recall Curve
Precision
Recall
Calibration Curve
Model
Perfectly calibrated
Fraction of Positives
Mean Predicted Probability
Confusion Matrix
True Label
Predicted Label
46121
46112


Mean Feature Importances (Instances 100-199)
Subject
Predicate
Object
0.3
0.00000
0.00002
0.00004
0.00006
0.00008
0.00010
0.00012
0.00014
Contribution to Prediction Probability

ComplEx


Training vs Validation Loss
Training Loss
Validation Loss
Loss
Epoch
Precision-Recall Curve
Precision
Recall
Calibration Curve
Model
Perfectly calibrated
Fraction of Positives
Mean Predicted Probability
Confusion Matrix
True Label
Predicted Label
46121
46112



	C1	C2	C3	C4	C5	C6	C7	C8
$TransE$	11	49	0	0	0.4726	0.7412	0.1389	0.1093
$TransH$	27	0	0	0	0.4295	0.5081	0.1932	0.0821
$RotatE$	31	0	0	0	0.1157	0.1293	0.1031	0.0071
$HoIE$	18	18	0	0	0.4174	0.6353	0.1006	0.1089
$DistMult$	1	1301	0	0	0.6798	0.7938	0.5669	0.0553
$ComplEx$	43	516	0	0	0.6005	0.6692	0.5706	0.0239

$TransE$

Top 5 most frequent relations:
[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/term', 633),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasURI', 461),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/content', 229),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/definition', 109),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/relatedTerm', 24)]
Relations appearing only once:
['https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasURL',
'https://ec.europa.eu/eurostat/NLP4StatRef/ontology/title',
'https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasCategoryOfGlossaryArticle']
Relation with highest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasURL', 0.5593205094337463)
Relation with lowest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/title', 0.3952298164367676)

$TransH$

Top 5 most frequent relations:
[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/definition', 1290),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/title', 27),

('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/sourcePublication', 27),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/term', 18),
('http://www.w3.org/1999/02/22-rdf-syntax-ns#first', 14)]
Relations appearing only once:
['http://www.w3.org/2000/01/rdf-schema#subPropertyOf',
'http://www.w3.org/2000/01/rdf-schema#domain',
'https://ec.europa.eu/eurostat/NLP4StatRef/ontology/databasePath']
Relation with highest average probability:
('http://www.w3.org/2002/07/owl#equivalentProperty', 0.4814907133579254)
Relation with lowest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasEurostatTheme',
0.35845009982585907)

$HoIE$

Top 5 most frequent relations:
[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasReference', 423),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/databasePath', 302),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/title', 143),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasParagraph', 141),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/relatedTerm', 134)]
Relations appearing only once:
['https://ec.europa.eu/eurostat/NLP4StatRef/ontology/term',
'https://ec.europa.eu/eurostat/NLP4StatRef/ontology/remark',
'http://www.w3.org/2002/07/owl#backwardCompatibleWith']
Relation with highest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/remark', 0.48419255018234253)
Relation with lowest average probability:
('http://www.w3.org/2002/07/owl#backwardCompatibleWith', 0.2759597897529602)

$DistMult$

Top 5 most frequent relations:
[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/fileLink', 1500)]
Relation with highest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/fileLink', 0.6797794284025828)
Relation with lowest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/fileLink', 0.6797794284025828)

$ComplEx$

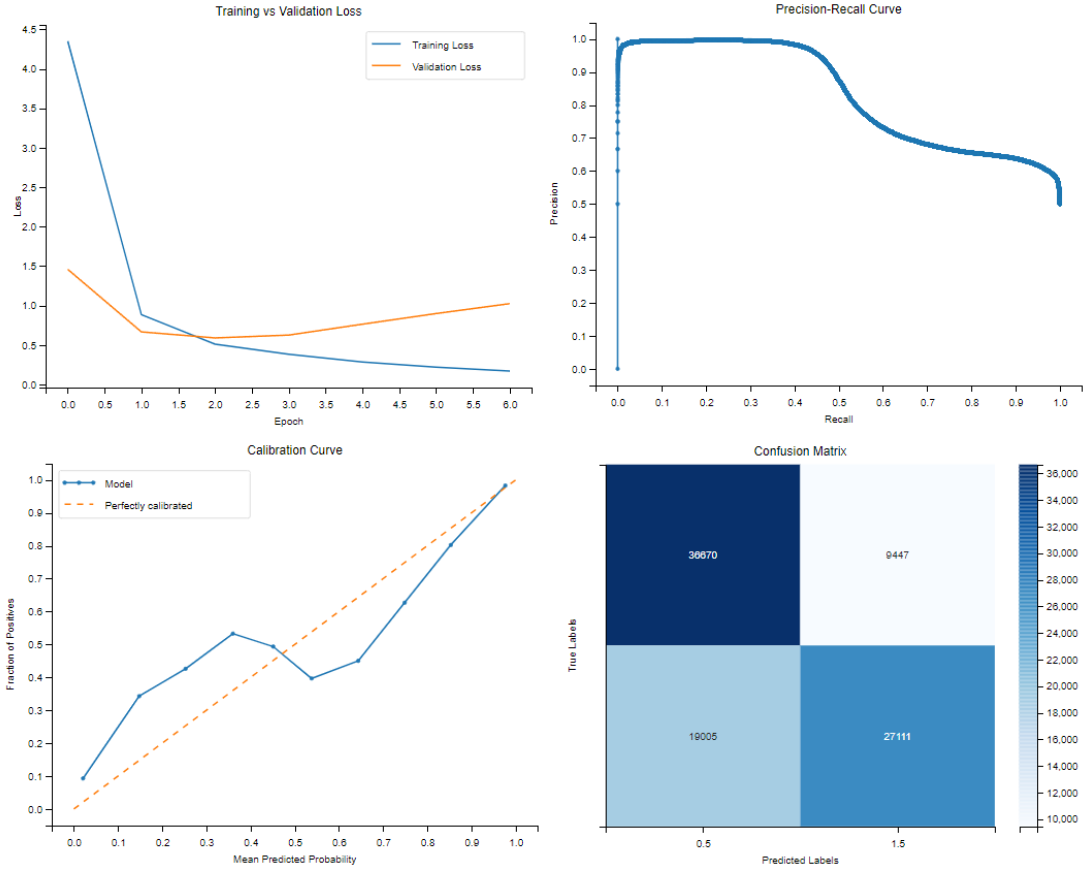
Top 5 most frequent relations:
[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/term', 70),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasEurostatTheme', 53),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/keyword', 48),

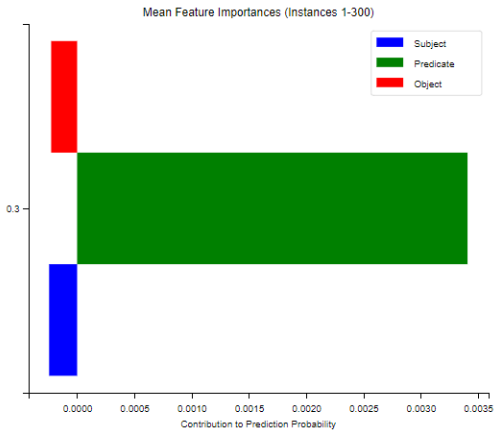
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasURI', 47),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasURL', 47)]
Relation with highest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/context', 0.6107146433421544)
Relation with lowest average probability:
('http://www.w3.org/1999/02/22-rdf-syntax-ns#first', 0.5913937477504506)

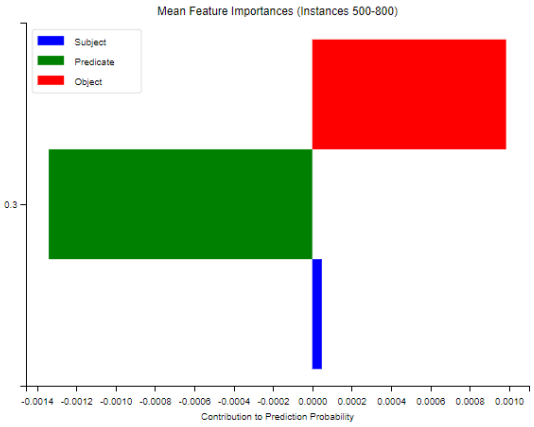
VII. Group 7

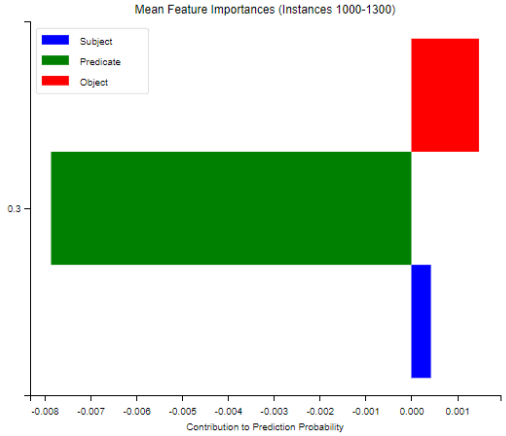
	Mean Silhouette	**Mean Davies-Bouldin**	**Mean Calinski-Harabasz**
$TransE$	0.8061	0.2486	3957.2206
$TransH$	0.7834	0.2856	3052.8489
$RotatE$	0.7912	0.2727	4089.8736
$NoEmbds$	**0.8055**	**0.2481**	**4085.5100**
$HoIE$	0.8087	0.2464	3820.3877
$DistMult$	0.7913	0.2738	4053.6681

$TransE$



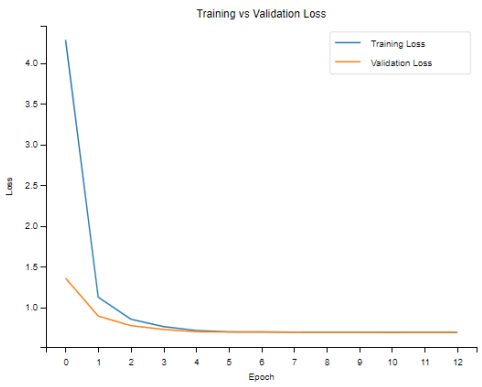


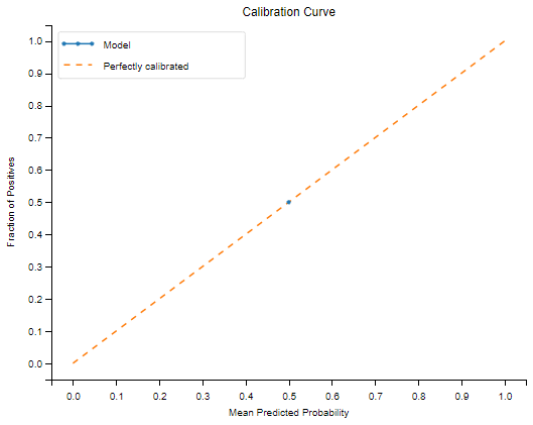




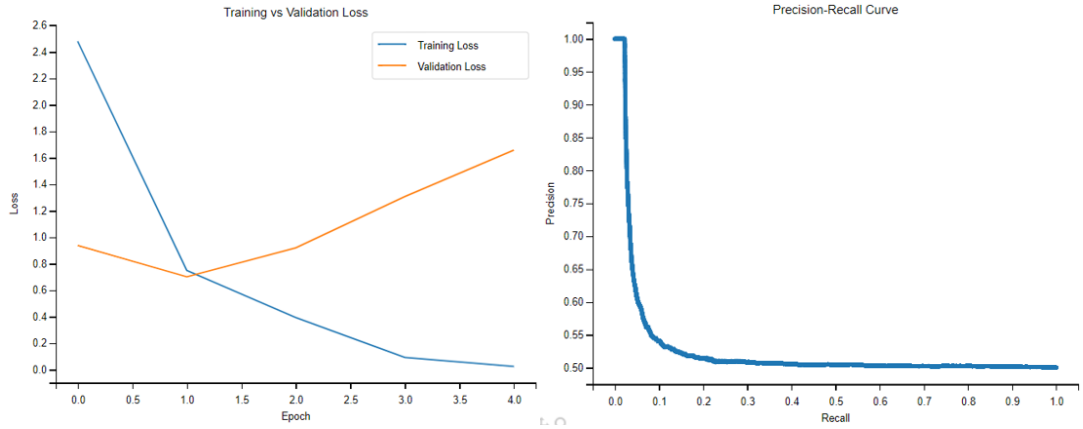
	subject	predicate	object
Triple 1	-0.0022	0.1603	0.0345
Triple 2	0.0372	0.0609	0.0050
Triple 3	0.0011	-0.39708	-0.0489
Triple 4	-0.0317	0.16032	0.0097
Triple 5	-0.0083	0.09255	0.0157

$TransH$

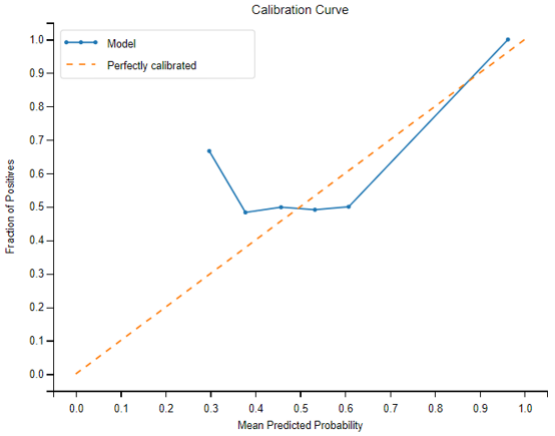


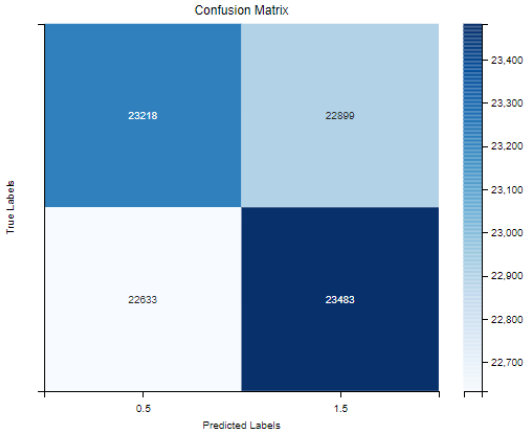


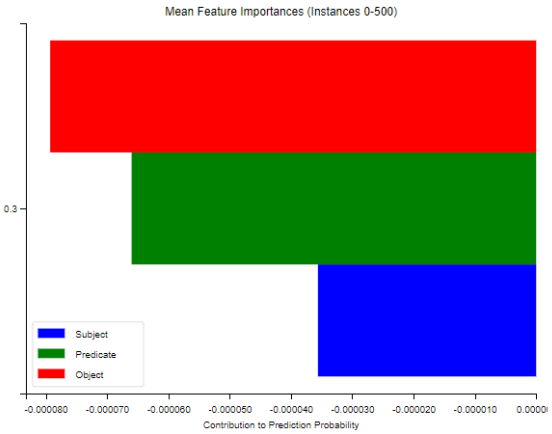
RotatE


Training vs Validation Loss
Training Loss
Validation Loss
Loss
Epoch

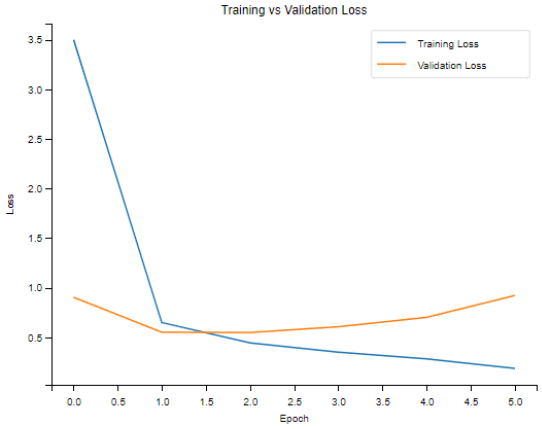
Precision-Recall Curve
Precision
Recall

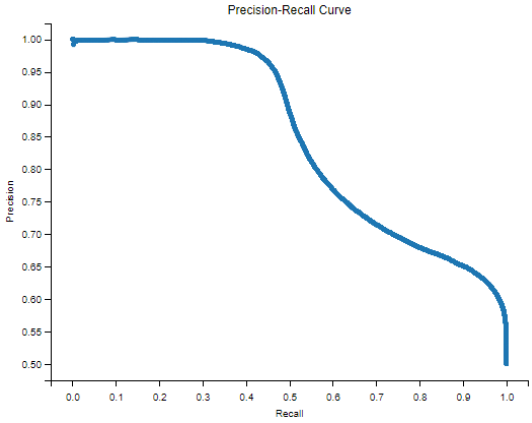

Calibration Curve
Model
Perfectly calibrated
Fraction of Positives
Mean Predicted Probability

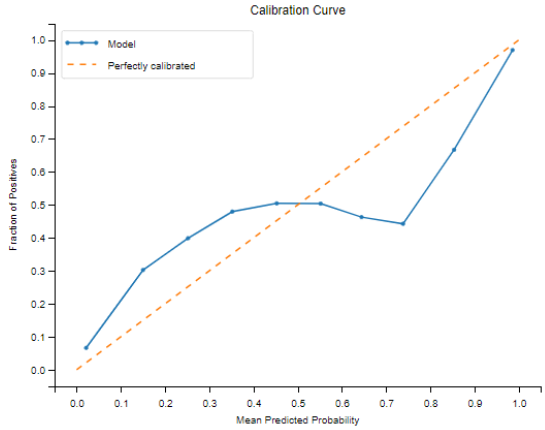

Confusion Matrix
23218
22899
22633
23483
True Labels
Predicted Labels

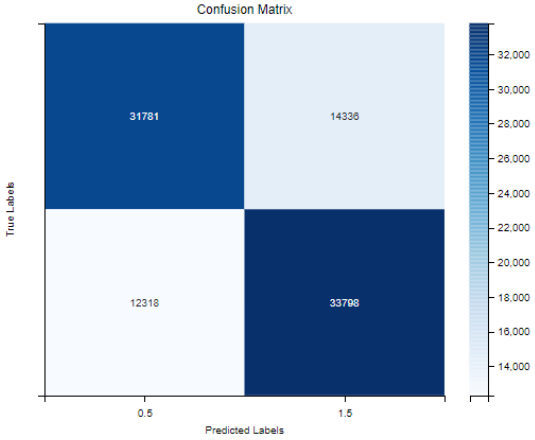

Mean Feature Importances (Instances 0-500)
Subject
Predicate
Object
Contribution to Prediction Probability

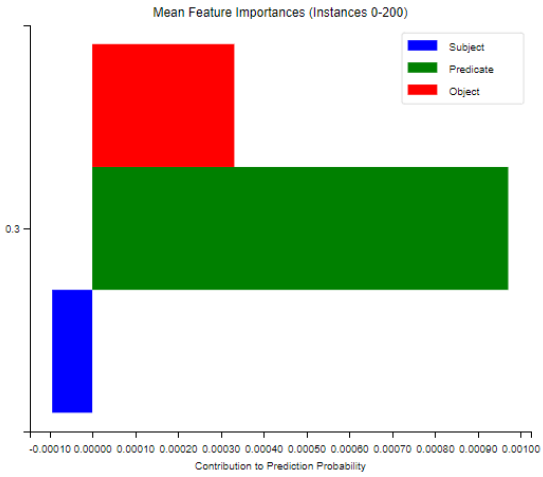
NoEmbds


Training vs Validation Loss
Training Loss
Validation Loss
Loss
Epoch

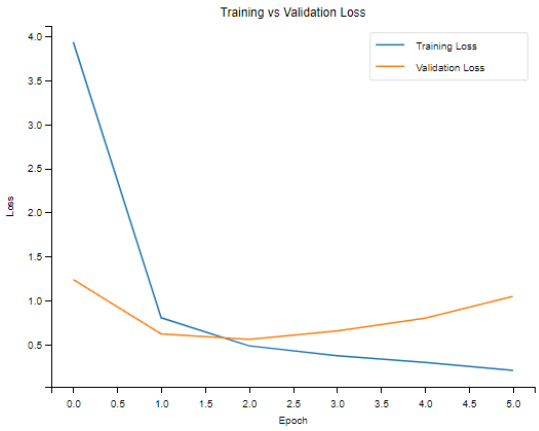

Precision-Recall Curve
Precision
Recall

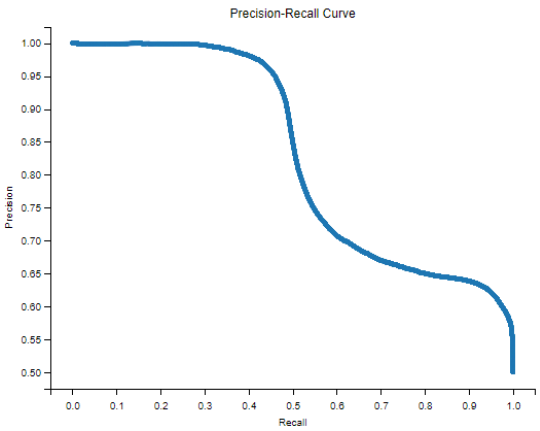

Calibration Curve
Model
Perfectly calibrated
Fraction of Positives
Mean Predicted Probability

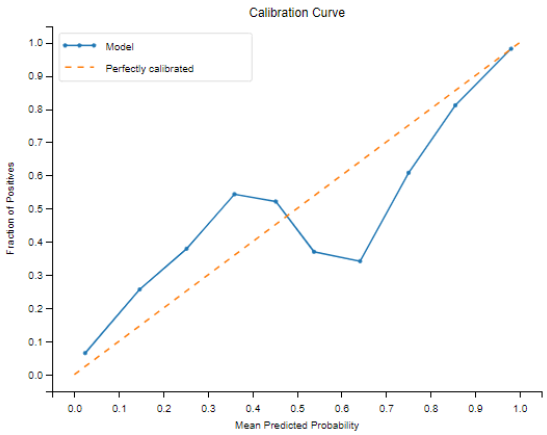

Confusion Matrix
31781
14336
12318
33798
True Labels
Predicted Labels

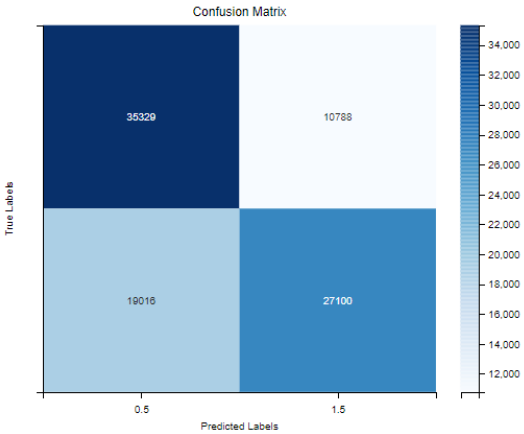

Mean Feature Importances (Instances 0-200)
Subject
Predicate
Object
Contribution to Prediction Probability

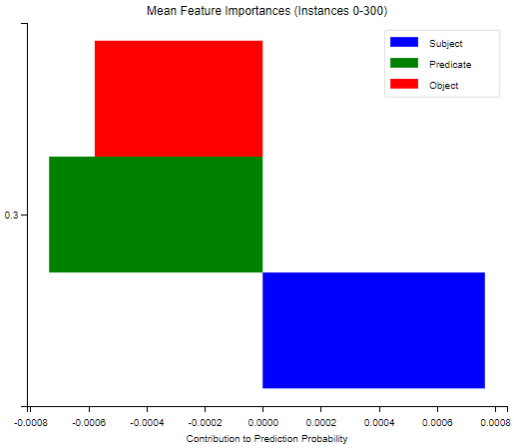
HoIE

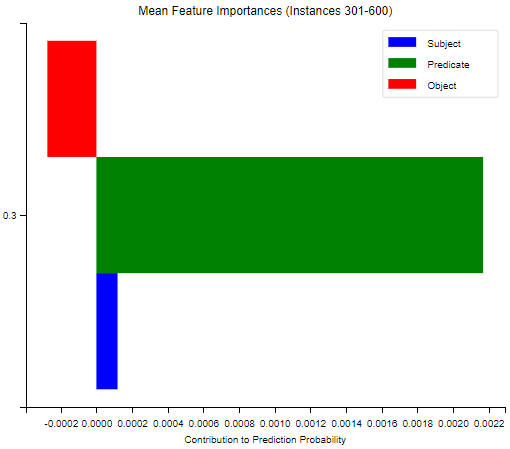

Training vs Validation Loss
Training Loss
Validation Loss
Loss
Epoch

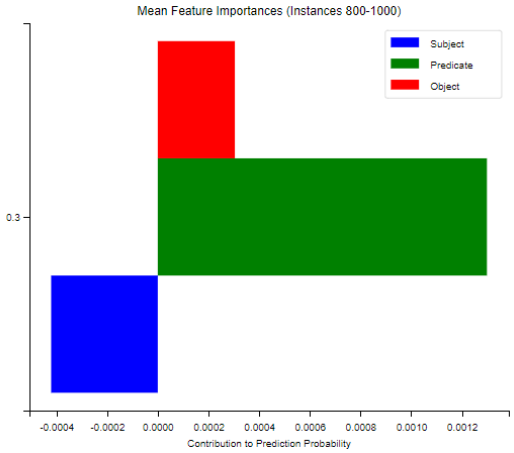

Precision-Recall Curve
Precision
Recall


Calibration Curve
Model
Perfectly calibrated
Fraction of Positives
Mean Predicted Probability

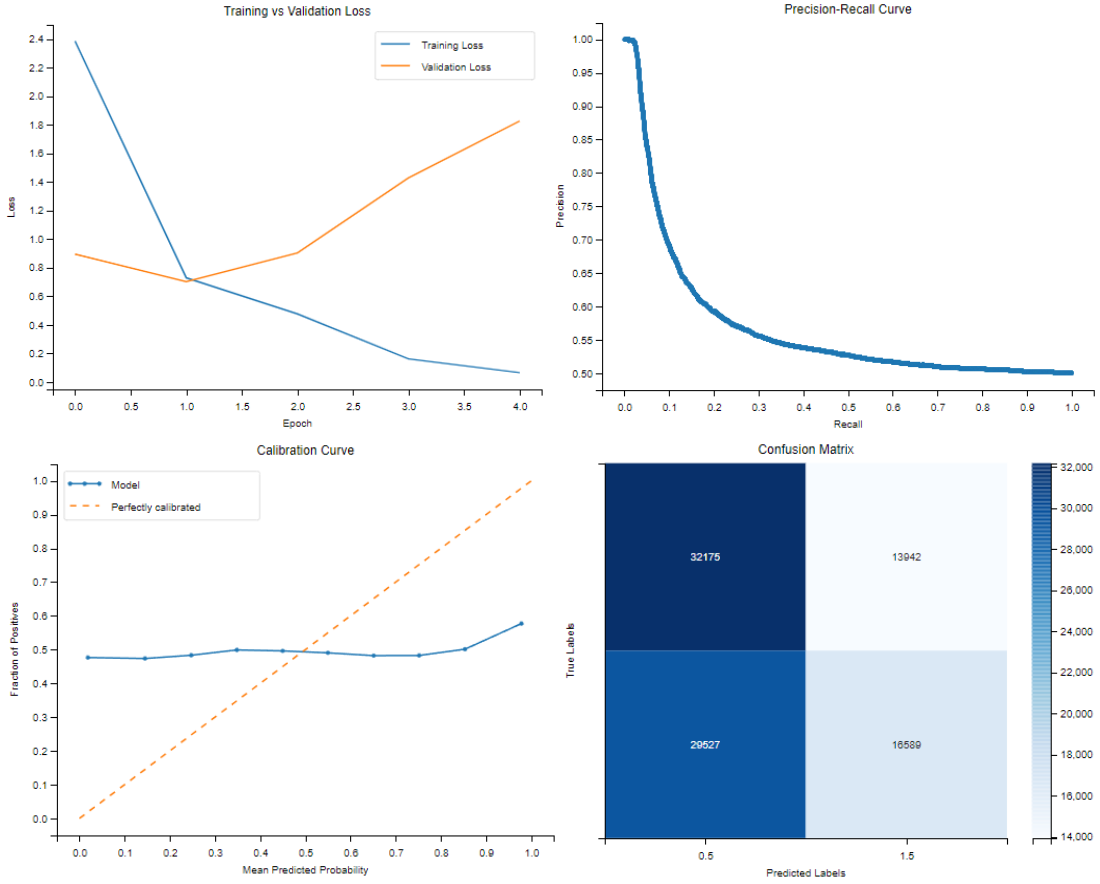

Confusion Matrix
35329
10788
19016
27100
True Labels
Predicted Labels

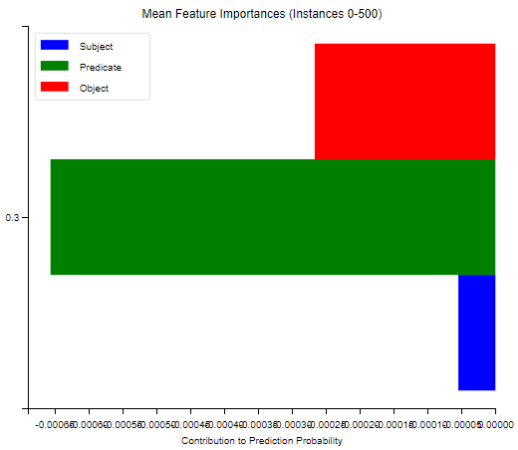

Mean Feature Importances (Instances 0-300)
Subject
Predicate
Object
Contribution to Prediction Probability

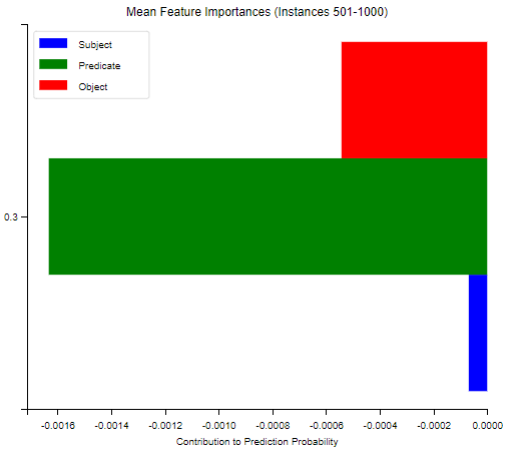

Mean Feature Importances (Instances 301-600)
Subject
Predicate
Object
Contribution to Prediction Probability

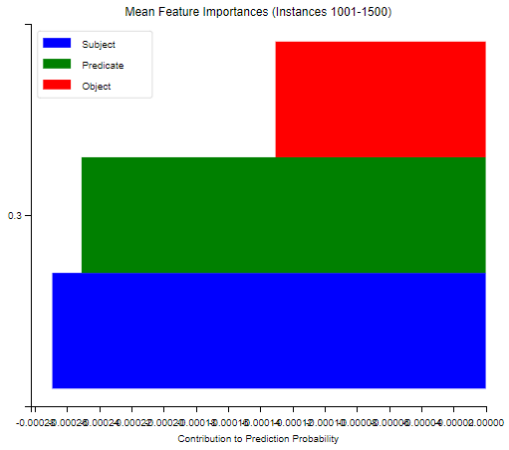

Mean Feature Importances (Instances 800-1000)
Subject
Predicate
Object
0.3
-0.0004
-0.0002
0.0000
0.0002
0.0004
0.0006
0.0008
0.0010
0.0012
Contribution to Prediction Probability

DistMult

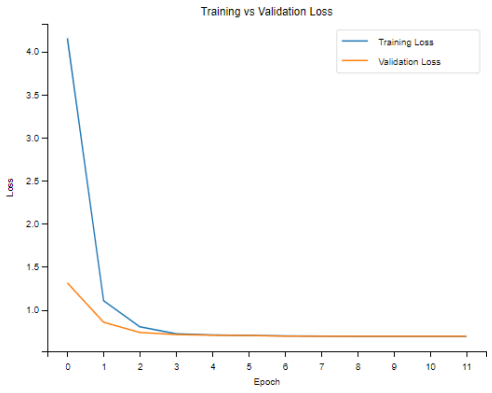

Training vs Validation Loss
Training Loss
Validation Loss
Loss
Epoch
Precision-Recall Curve
Precision
Recall
Calibration Curve
Model
Perfectly calibrated
Fraction of Positives
Mean Predicted Probability
Confusion Matrix
32175
13942
29527
16589
True Labels
Predicted Labels

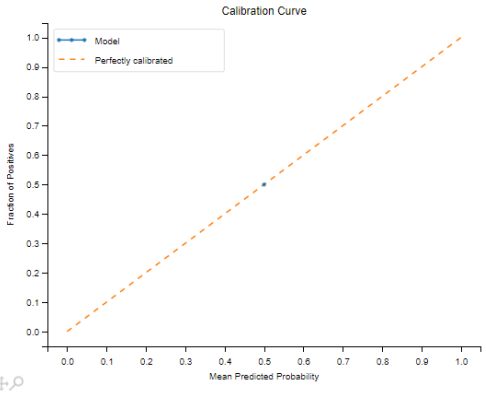






ComplEx





	C1	C2	C3	C4	C5	C6	C7	C8
TransE	13	76	6	3	0.3128	0.9611	0.0002	0.2202
TransH	1	0	0	0	0.5000	0.5000	0.5000	0
RotatE	43	0	0	0	0.4872	0.5781	0.2688	0.0648
NoEmbds	18	290	32	9	0.3319	0.9979	0.0002	0.2682
HoIE	10	97	0	0	0.2909	0.8395	0.0002	0.2169
DistMult	43	427	256	43	0.3203	0.9996	0.0019	0.3863

$TransE$

Top 5 most frequent relations:
[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/dataSource', 584),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/term', 294),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasCode', 232),
('http://www.w3.org/2000/01/rdf-schema#subClassOf', 160),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/context', 133)]
Relations appearing only once:
['https://ec.europa.eu/eurostat/NLP4StatRef/ontology/title',
'http://www.w3.org/2000/01/rdf-schema#comment',
'https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasURI']
Relation with highest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/content', 0.5114988520120581)
Relation with lowest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasURI', 0.0035236880648881197)

$RotatE$

Top 5 most frequent relations:
[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasCode', 99),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasOECDTheme', 51),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasURL', 42),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasURI', 41),
('http://www.w3.org/2000/01/rdf-schema#range', 41)]
Relation with highest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasCategoryOfStatisticExplainedArticle', 0.5164193113644918)
Relation with lowest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/level', 0.4562055060157069)

$NoEmbds$

Top 5 most frequent relations:
[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasCode', 640),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/term', 158),
('http://www.w3.org/2000/01/rdf-schema#subPropertyOf', 130),
('http://www.w3.org/2002/07/owl#equivalentProperty', 107),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasReference', 89)]
Relations appearing only once:
['http://www.w3.org/2000/01/rdf-schema#range',
'https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasURL']
Relation with highest average probability:
('http://www.w3.org/2000/01/rdf-schema#range', 0.4767267107963562)

Relation with lowest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasURL', 0.003997777123004198)

$HoIE$

Top 5 most frequent relations:
[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/definition', 615),
('http://www.w3.org/2000/01/rdf-schema#label', 476),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/relatedTerm', 216),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasURI', 88),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/context', 44)]
Relations appearing only once:
['http://www.w3.org/1999/02/22-rdf-syntax-ns#type']
Relation with highest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasURL', 0.34485906522798665)
Relation with lowest average probability:
('http://www.w3.org/1999/02/22-rdf-syntax-ns#type', 0.0012193239526823163)

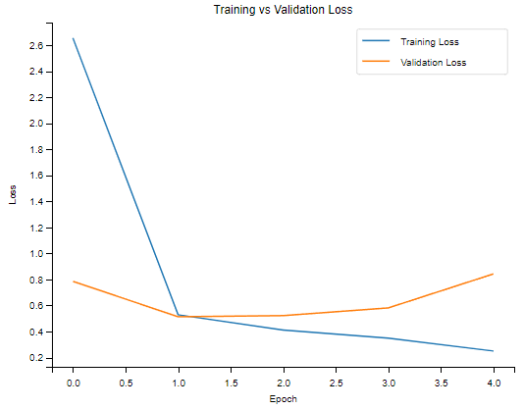
$DistMult$

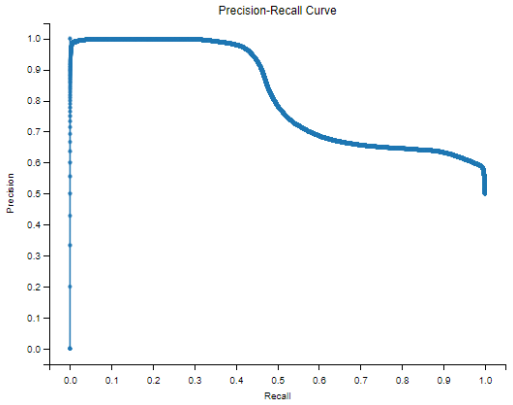
Top 5 most frequent relations:
[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasTopic', 59),
('http://www.w3.org/1999/02/22-rdf-syntax-ns#type', 58),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasOECDTheme', 49),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/relatedTerm', 48),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasReference', 46)]
Relation with highest average probability:
('http://www.w3.org/2000/01/rdf-schema#comment', 0.5469163116857609)
Relation with lowest average probability:
('http://www.w3.org/2002/07/owl#backwardCompatibleWith', 0.16579647843665893)

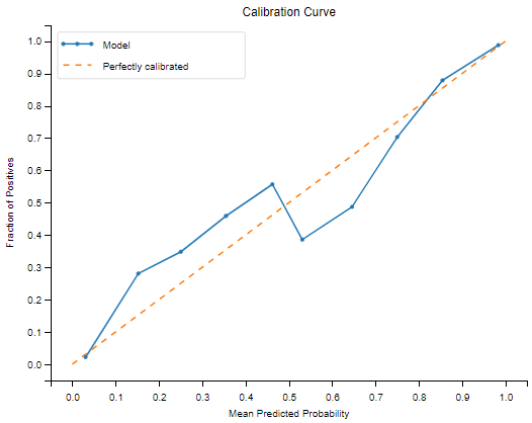
VIII. Group 8

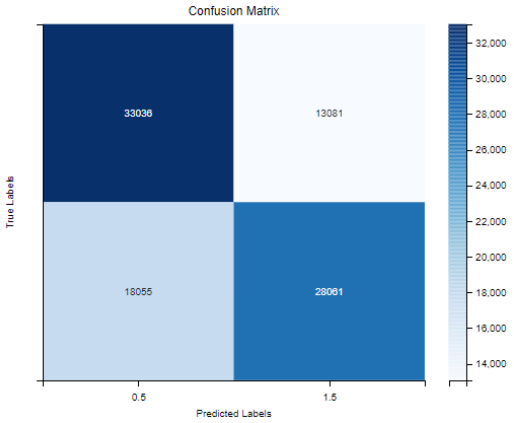
	Mean Silhouette	**Mean Davies-Bouldin**	**Mean Calinski-Harabasz**
$TransE$	0.8091	0.2447	3706.9005
$TransH$	0.7838	0.2843	3051.8689
$RotatE$	0.8056	0.2494	3804.2665
$NoEmbds$	**0.8117**	**0.2402**	**3664.2924**
$HoIE$	0.7949	0.2663	4074.5816
$DistMult$	0.7961	0.2659	4306.1863

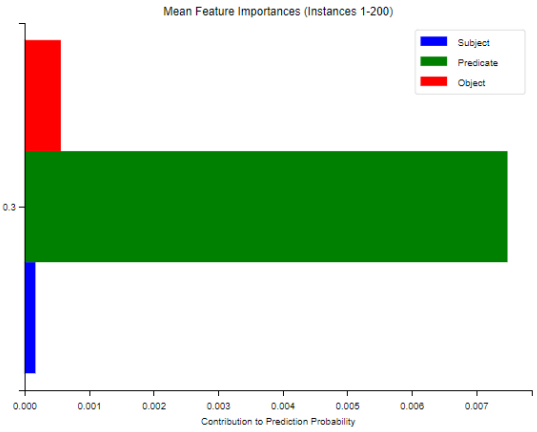
TransE

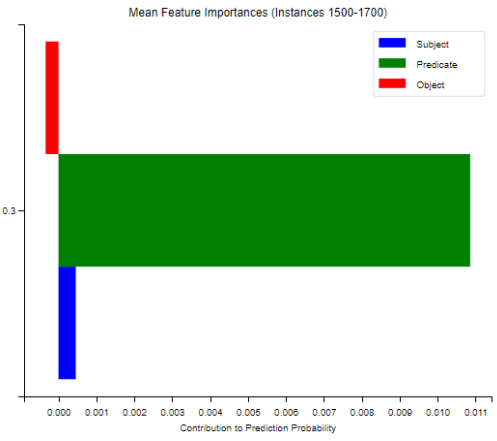


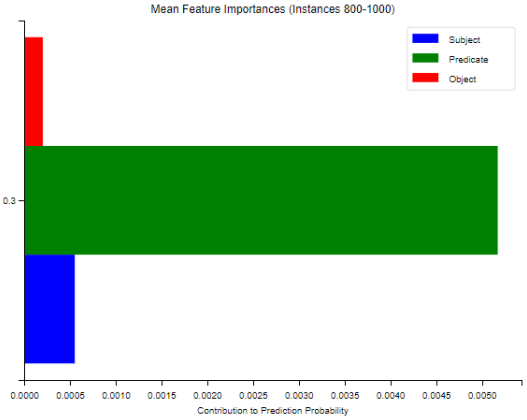






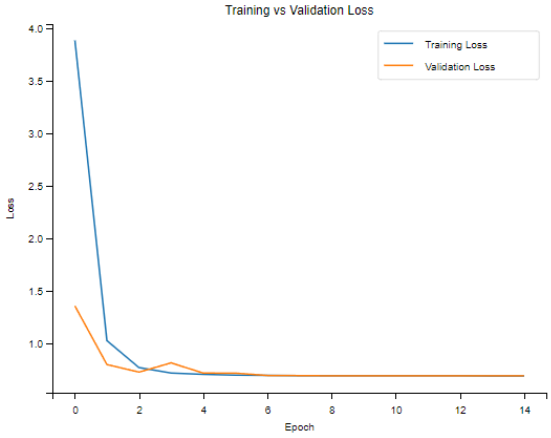


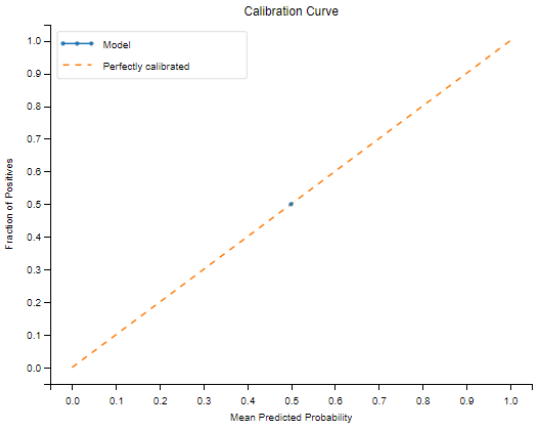




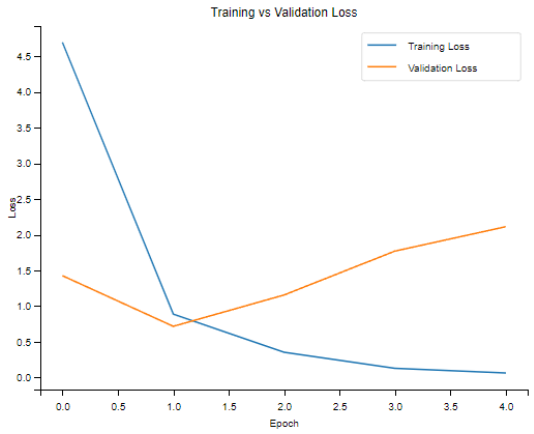
	subject	**predicate**	**object**
Triple 1	-0.0042	0.011997	0.0260
Triple 2	-0.0050	0.015704	0.0244
Triple 3	-0.0023	-0.02397	-0.0660
Triple 4	0.0045	-0.02554	0.0238
Triple 5	0.0042	0.01416	-0.0033

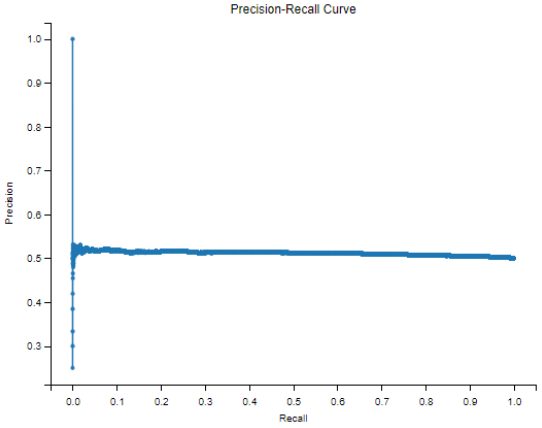
$TransH$

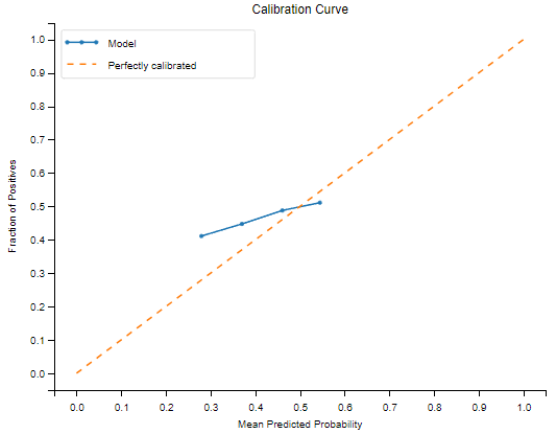


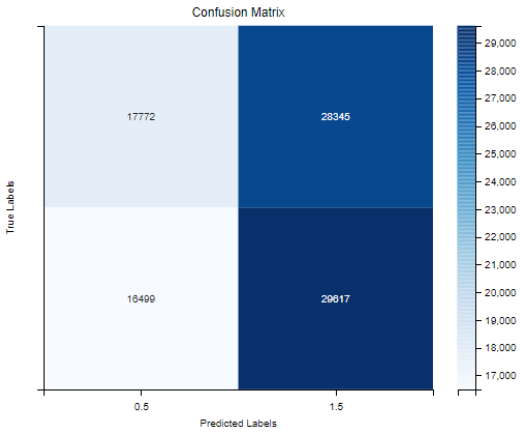


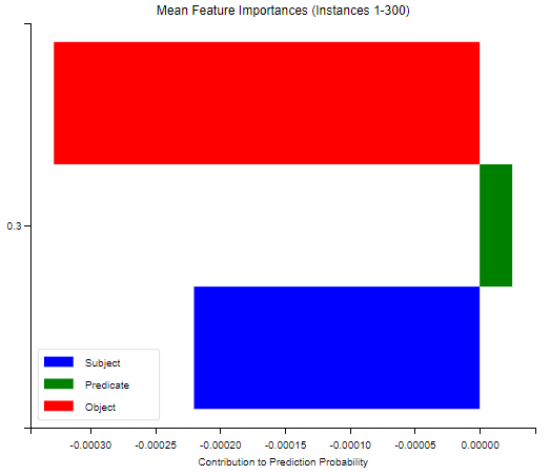
$RotatE$

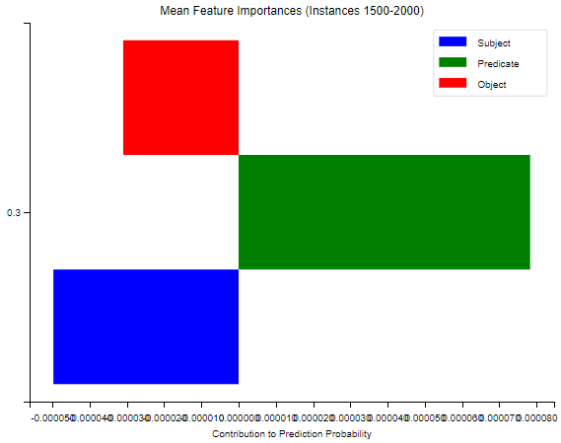






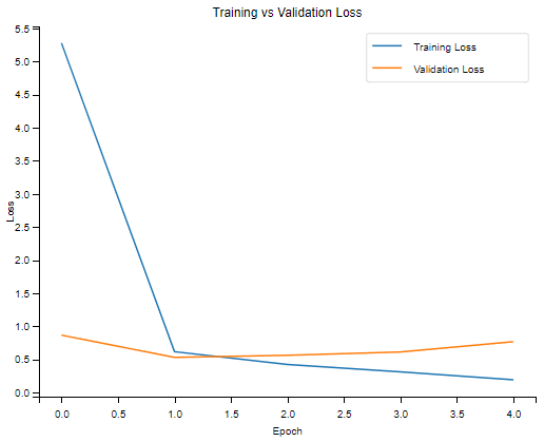


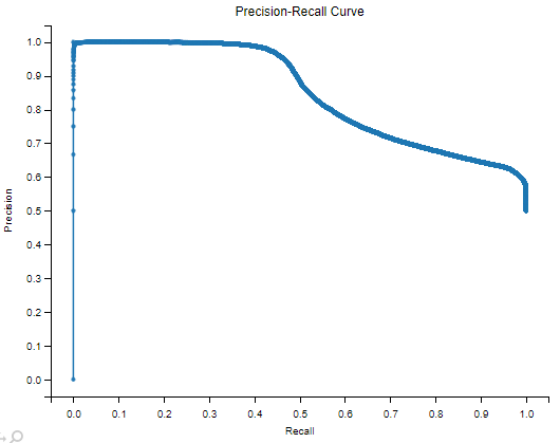


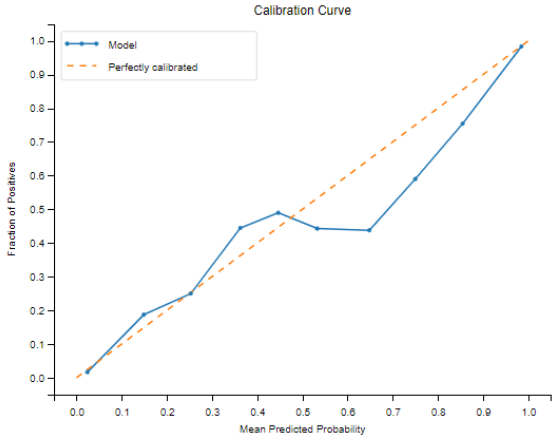


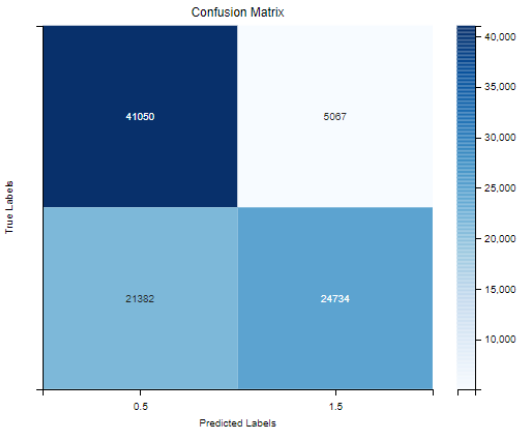
	subject	predicate	object
Triple 1	-0.0290	-0.00805	-0.0633
Triple 2	0.0185	0.01967	0.0116
Triple 3	0.0192	-0.0037	0.0274
Triple 4	0.0028	-0.01595	0.0103
Triple 5	-0.0086	-0.00528	0.0126

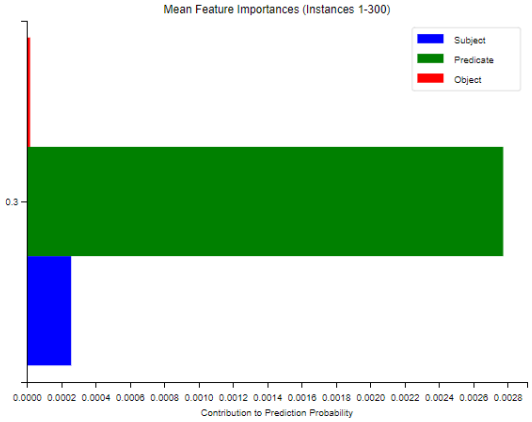
NoEmbds

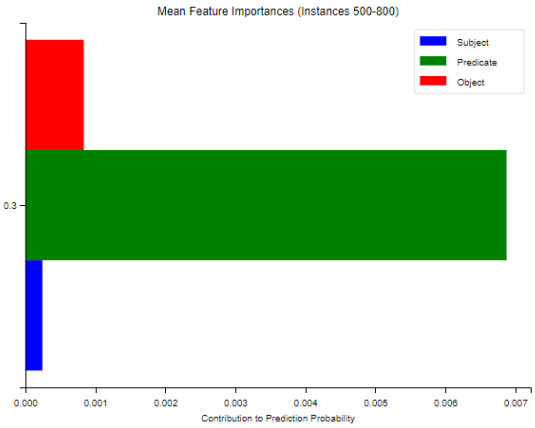


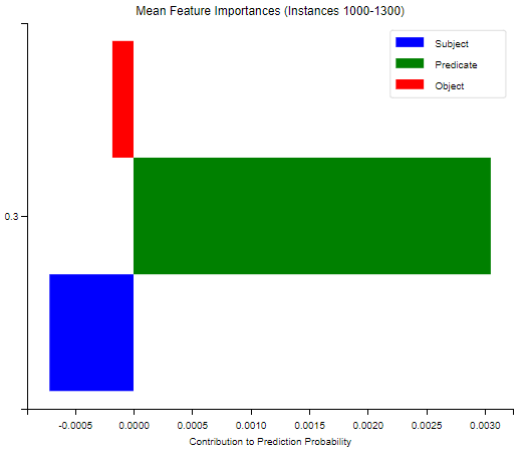






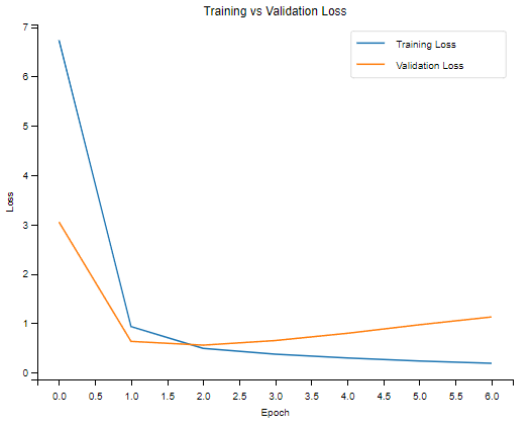


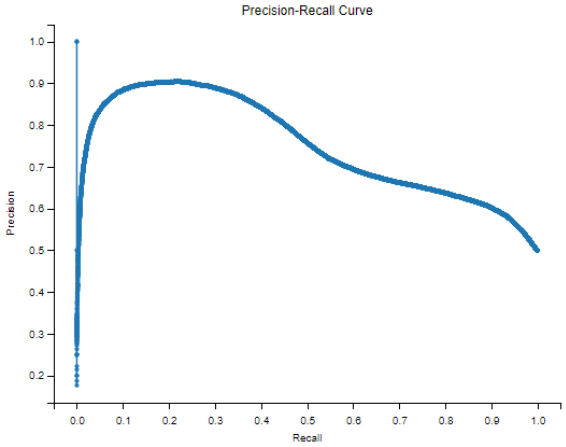


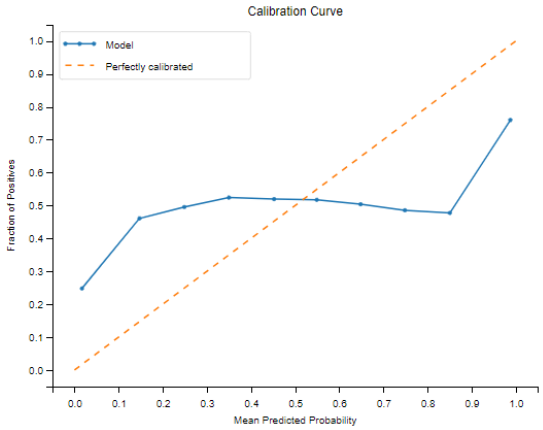


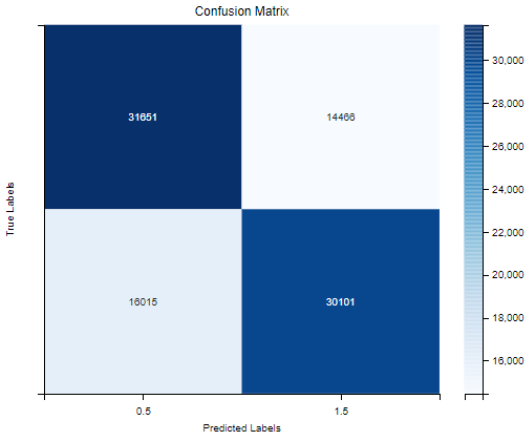
	subject	predicate	object
Triple 1	0.0486	-0.22033	0.0050
Triple 2	-0.0026	0.32080	0.0464
Triple 3	0.0548	-0.06042	0.0399
Triple 4	-0.0917	-0.07907	-0.1146
Triple 5	0.0199	-0.06625	0.0363

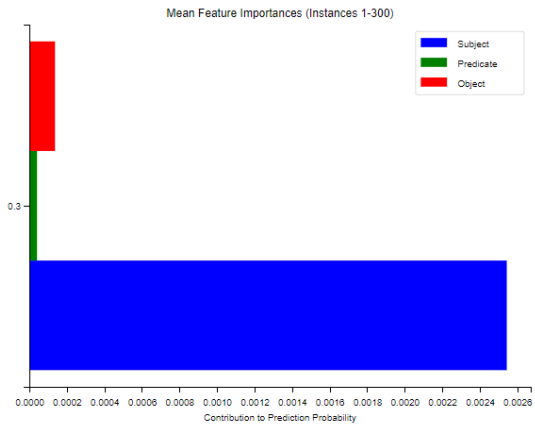
HoIE

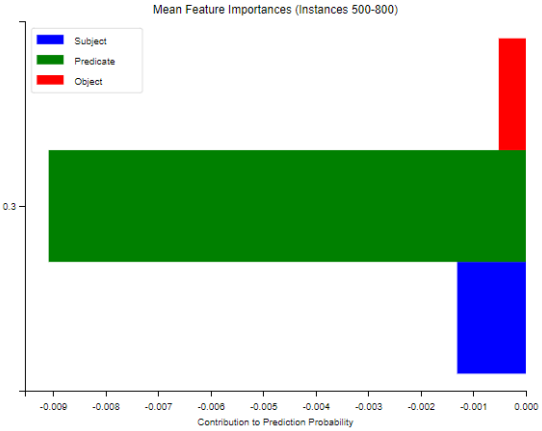

Training vs Validation Loss
Training Loss
Validation Loss
Loss
Epoch

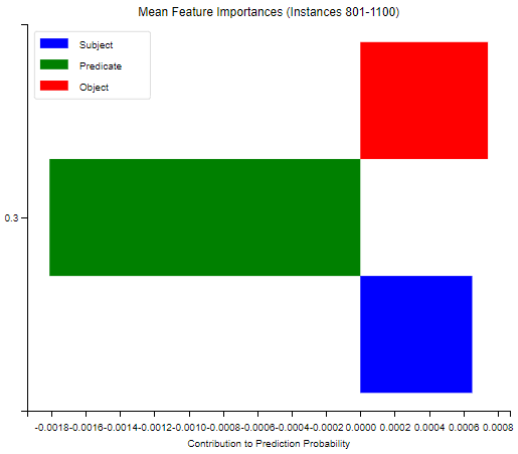

Precision-Recall Curve
Precision
Recall


Calibration Curve
Model
Perfectly calibrated
Fraction of Positives
Mean Predicted Probability


Confusion Matrix
31651
14466
16015
30101
True Labels
Predicted Labels

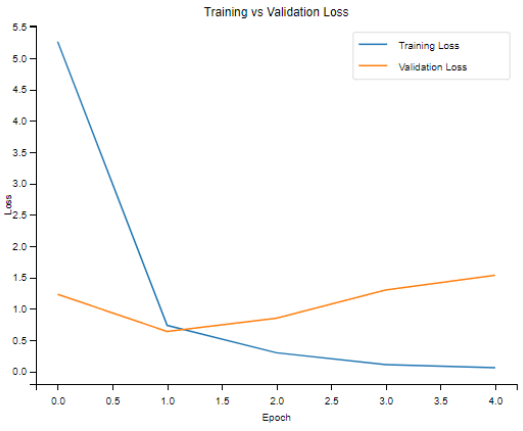

Mean Feature Importances (Instances 1-300)
Subject
Predicate
Object
Contribution to Prediction Probability

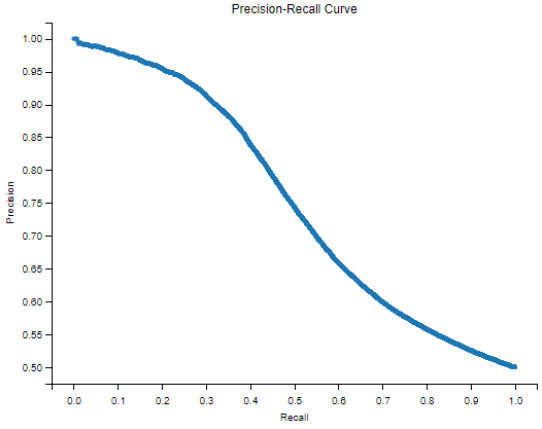

Mean Feature Importances (Instances 500-800)
Subject
Predicate
Object
Contribution to Prediction Probability

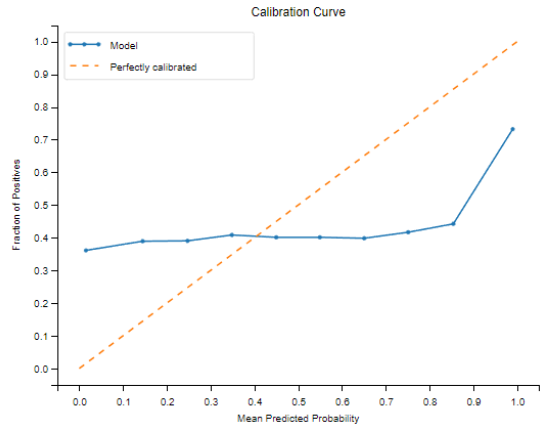


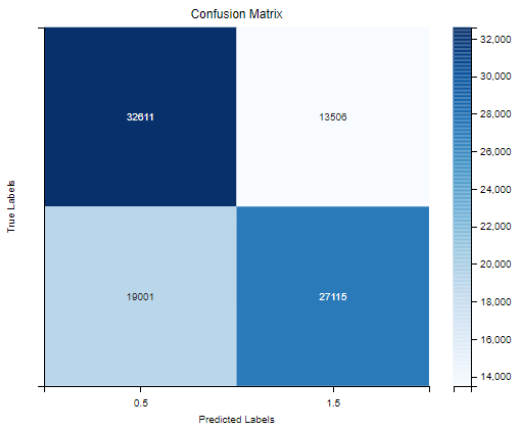
	subject	predicate	object
Triple 1	-0.0254	0.36054	-0.2732
Triple 2	0.0041	-0.48278	0.0560
Triple 3	-0.0383	-0.12850	0.1202
Triple 4	0.0109	0.42805	0.1681
Triple 5	-0.0039	0.01644	0.0384

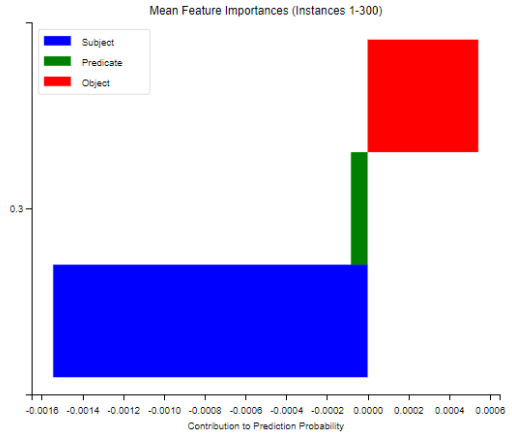
DistMult

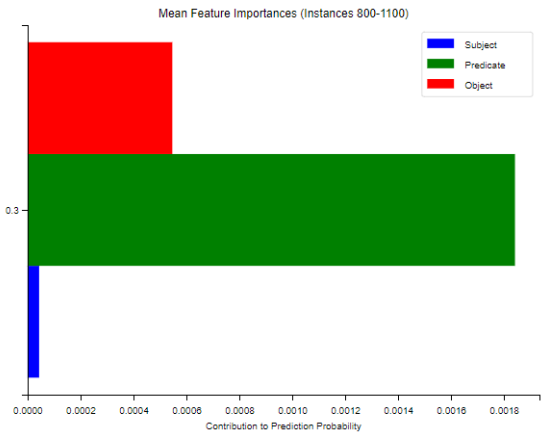


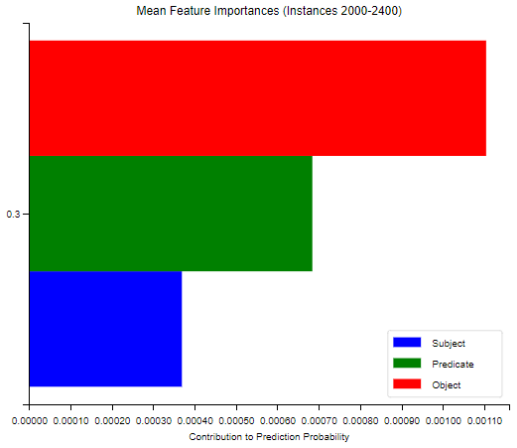












	subject	predicate	object
Triple 1	0.0246	0.24386	-0.0540
Triple 2	-0.0368	0.04157	0.2504
Triple 3	-0.0508	0.18885	-0.0265
Triple 4	0.0183	-0.2265	-0.0885
Triple 5	-0.0510	-0.2191	-0.1428

	C1	C2	C3	C4	C5	C6	C7	C8
$TransE$	11	132	7	2	0.3574	0.9526	0.0015	0.2216
$TransH$	1	0	0	0	0.5	0.5	0.5	0
$RotatE$	4	0	0	0	0.5067	0.5729	0.2656	0.0579
$NoEmbds$	4	193	48	2	0.3518	0.9928	0.0011	0.2427
$HoIE$	31	501	249	21	0.3970	0.9999	0.0001	0.3726
$DistMult$	43	360	241	39	0.2813	1.0000	0.0013	0.3796

$TransE$

Top 5 most frequent relations:

[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/databasePath', 880),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/id', 255),
('http://www.w3.org/2000/01/rdf-schema#comment', 153),
('http://www.w3.org/2000/01/rdf-schema#subClassOf', 105),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasURL', 59)]

Relations appearing only once:

['https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasParagraph',
'https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasCode',

'http://www.w3.org/2000/01/rdf-schema#label',
'https://ec.europa.eu/eurostat/NLP4StatRef/ontology/relatedTerm']
Relation with highest average probability:
('http://www.w3.org/2000/01/rdf-schema#label', 0.5816141963005066)
Relation with lowest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasParagraph',
0.003941344562917948)

$RotatE$

Top 5 most frequent relations:
[('http://www.w3.org/1999/02/22-rdf-syntax-ns#type', 397),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/term', 375),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasReference', 366),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/level', 362)
Relation with highest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/term', 0.5089655706882477)
Relation with lowest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasReference',
0.5016641690105689)

$NoEmbds$

Top 5 most frequent relations:
[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/level', 1040),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/term', 445),
('http://www.w3.org/1999/02/22-rdf-syntax-ns#type', 11),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasReference', 4)]
Relation with highest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasReference',
0.5648228228092194)
Relation with lowest average probability:
('http://www.w3.org/1999/02/22-rdf-syntax-ns#type', 0.2560354122384028)

$HoIE$

Top 5 most frequent relations:
[('http://www.w3.org/2002/07/owl#backwardCompatibleWith', 206),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasParagraph', 175),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/dataSource', 165),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasCode', 139),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/fileLink', 126)]
Relations appearing only once:
['https://ec.europa.eu/eurostat/NLP4StatRef/ontology/keyword',
'http://www.w3.org/2002/07/owl#equivalentProperty',

'http://www.w3.org/2000/01/rdf-schema#label',
'https://ec.europa.eu/eurostat/NLP4StatRef/ontology/context',
'http://www.w3.org/2000/01/rdf-schema#comment',
'http://www.w3.org/2002/07/owl#imports']
Relation with highest average probability:
('http://www.w3.org/2000/01/rdf-schema#label', 0.9927085041999817)
Relation with lowest average probability:
('http://www.w3.org/2002/07/owl#equivalentProperty', 0.0012489722575992346)

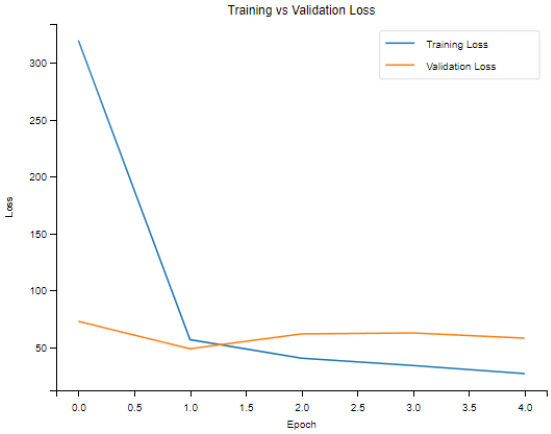
$DistMult$

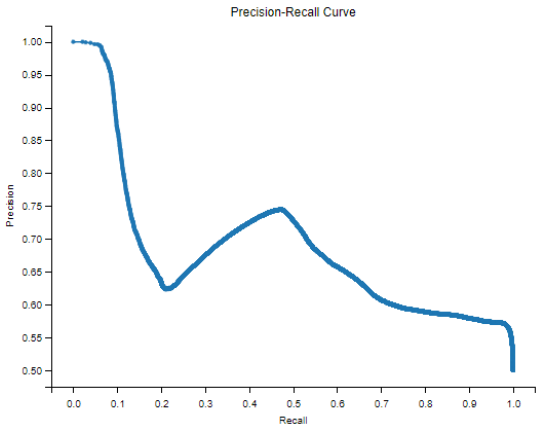
Top 5 most frequent relations:
[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasReference', 90),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/databasePath', 80),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/id', 80),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/title', 79),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasURL', 78)]
Relation with highest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/remark', 0.5627538235328923)
Relation with lowest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasCode', 0.06724924116861075)

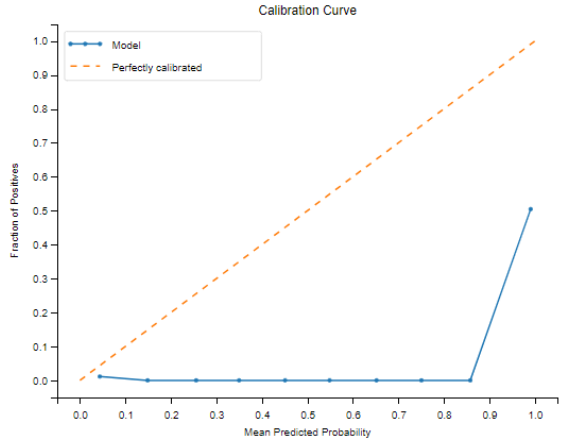
IX. Group 9

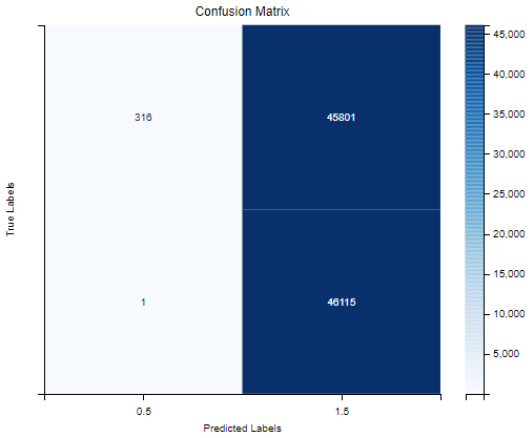
	Mean Silhouette	**Mean Davies-Bouldin**	**Mean Calinski-Harabasz**
$TransE$	0.8156	0.2357	3820.1842
$TransH$	0.7869	0.2806	3109.6894
$RotatE$	0.8088	0.2457	3977.7726
$NoEmbds$	**0.8168**	**0.2351**	**3893.3948**
$HoIE$	0.8110	0.2450	3699.0573
$DistMult$	0.7966	0.2621	3555.6123

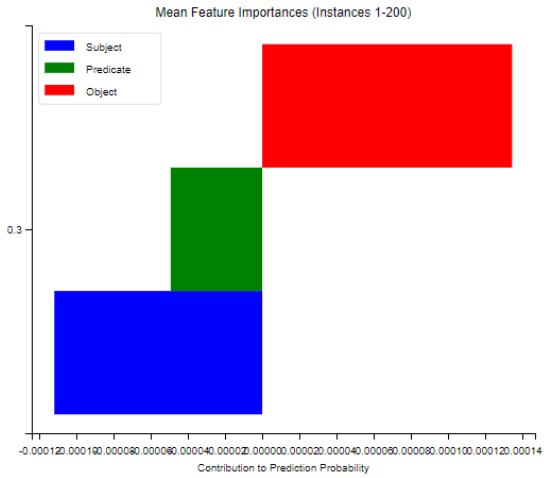
TransE

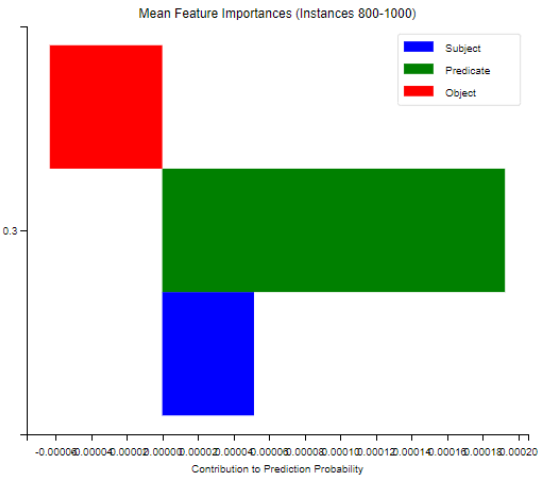

Training vs Validation Loss
Training Loss
Validation Loss
Loss
Epoch

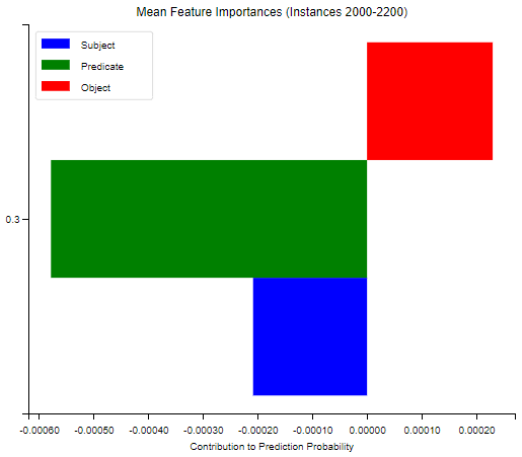

Precision-Recall Curve
Precision
Recall


Calibration Curve
Model
Perfectly calibrated
Fraction of Positives
Mean Predicted Probability


Confusion Matrix
316
45801
1
46115
True Labels
Predicted Labels

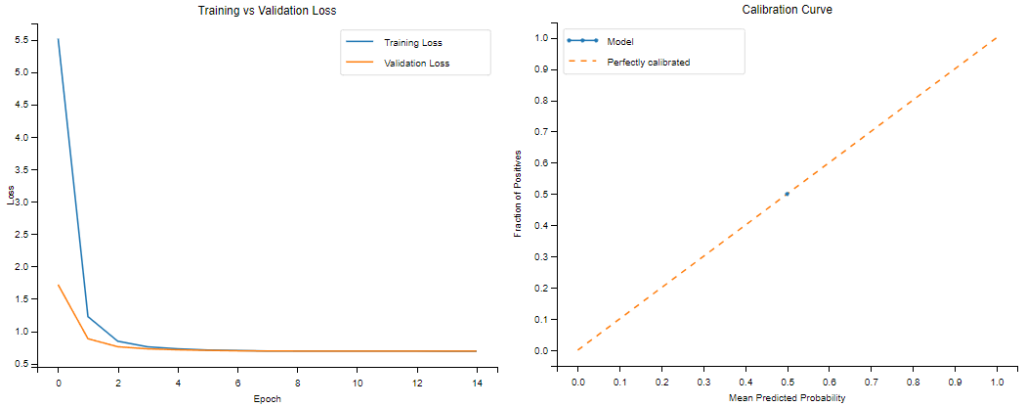

Mean Feature Importances (Instances 1-200)
Subject
Predicate
Object
Contribution to Prediction Probability


Mean Feature Importances (Instances 800-1000)
Subject
Predicate
Object
Contribution to Prediction Probability

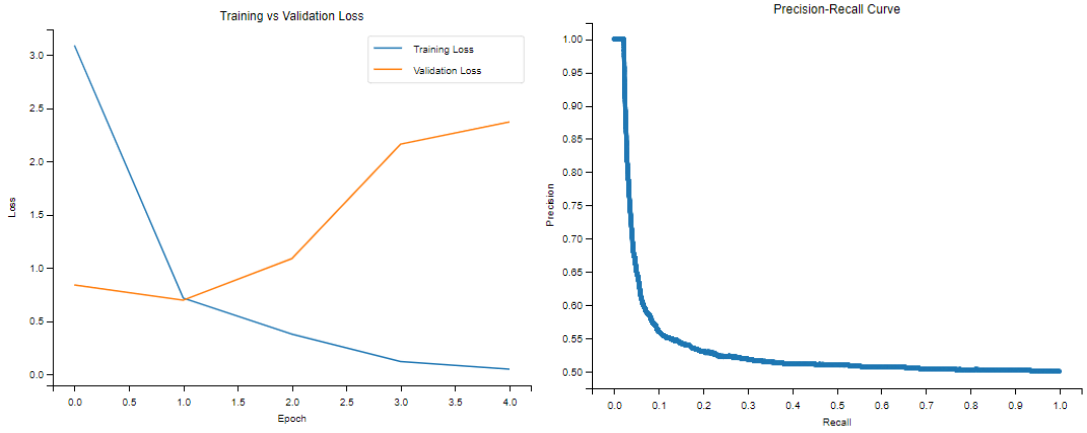


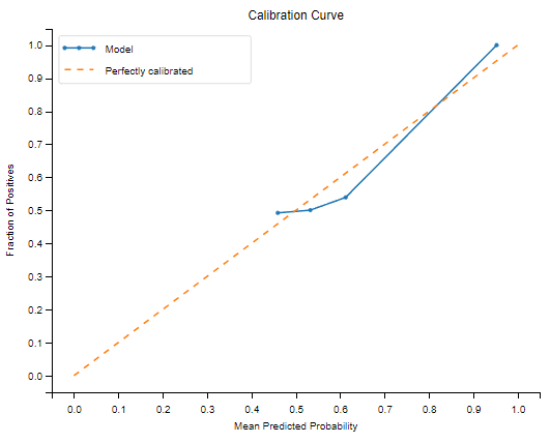
	subject	predicate	object
Triple 1	-0.0005	0.00137	-0.0005
Triple 2	-0.0005	0.00121	0.0020
Triple 3	0.0002	-0.0014	-0.0011
Triple 4	-6.2788	0.00127	-0.0013
Triple 5	0.0005	-0.0005	0.0020

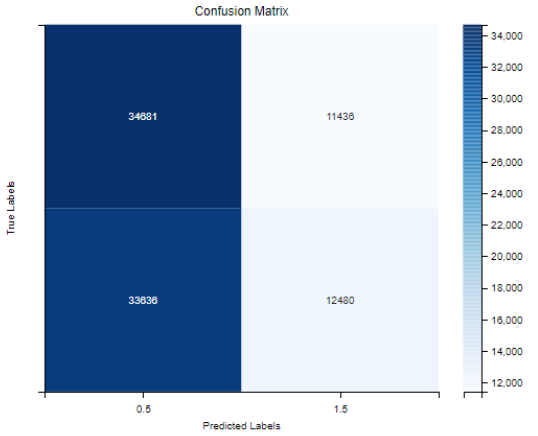
$TransH$

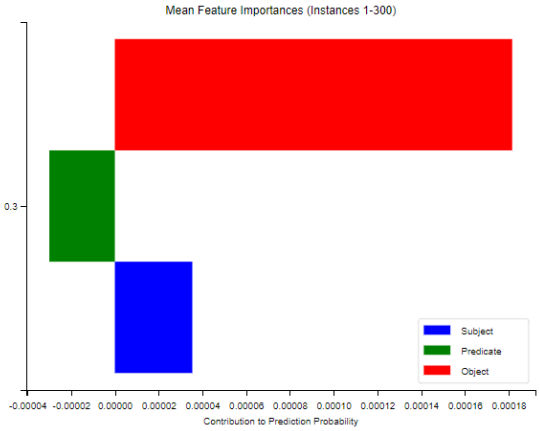


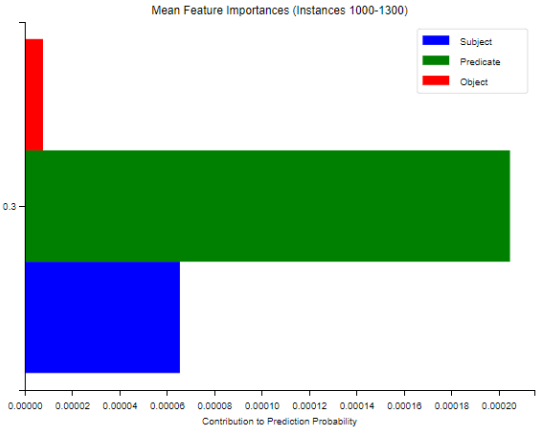
$RotatE$

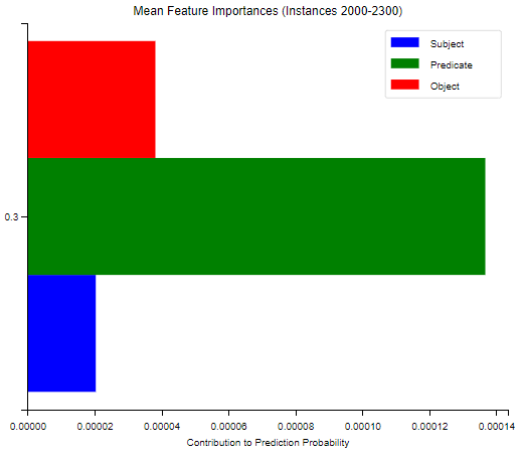






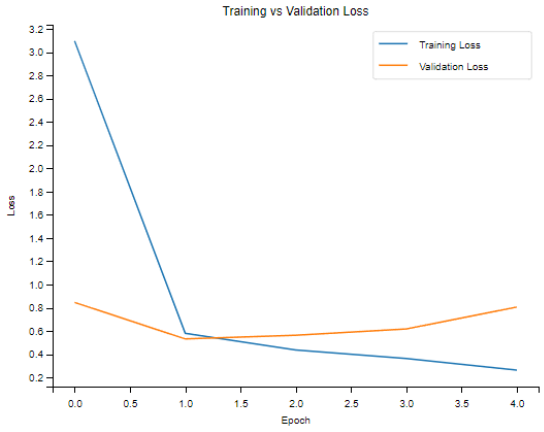


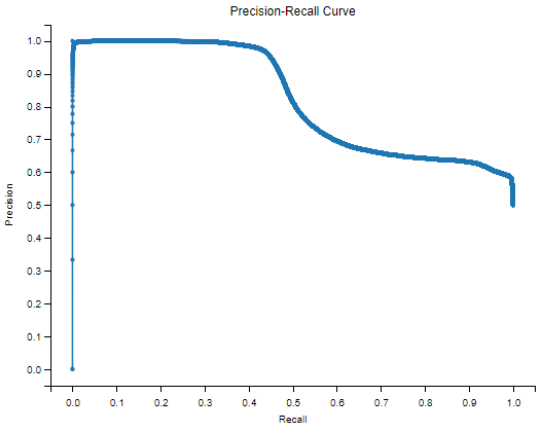


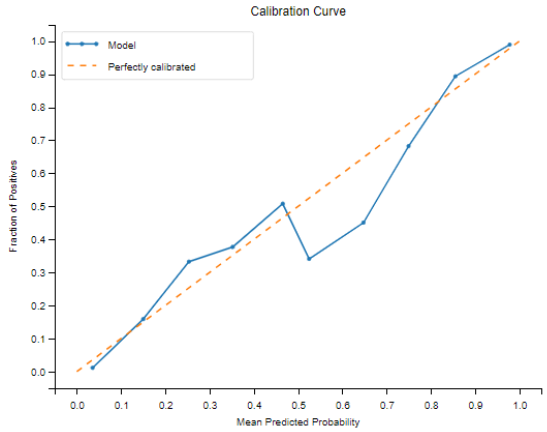


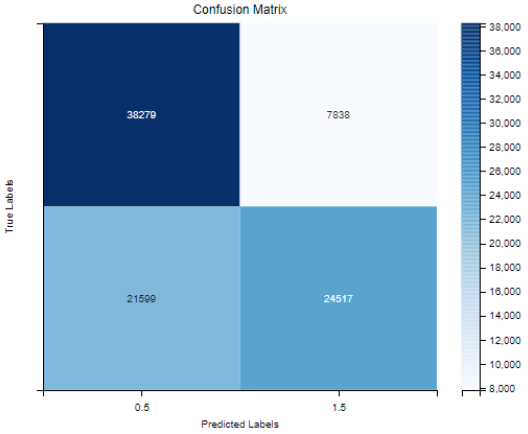
	subject	predicate	object
Triple 1	-0.0058	0.011330	-0.0055
Triple 2	-0.0080	0.010725	-0.0033
Triple 3	0.0125	-0.01959	-0.0025
Triple 4	-0.0045	0.01120	0.0011
Triple 5	0.0101	-0.02009	0.0016

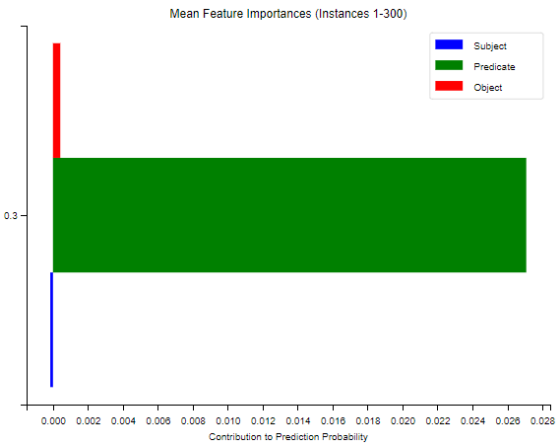
NoEmbds

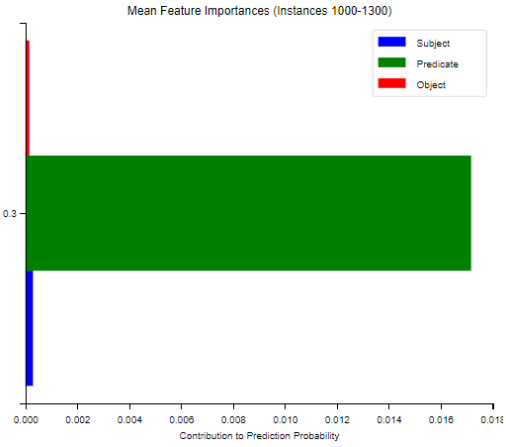

Training vs Validation Loss
Training Loss
Validation Loss
Loss
Epoch

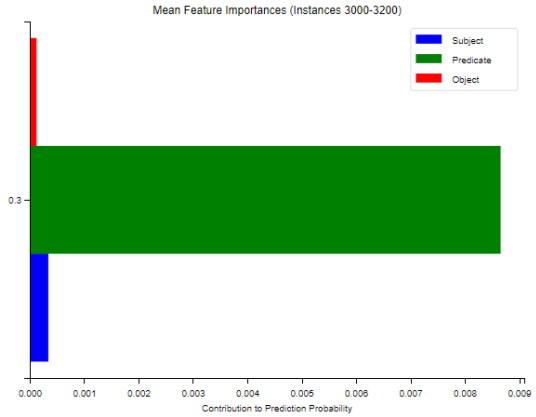

Precision-Recall Curve
Precision
Recall


Calibration Curve
Model
Perfectly calibrated
Fraction of Positives
Mean Predicted Probability


Confusion Matrix
38279
7838
21599
24517
True Labels
Predicted Labels

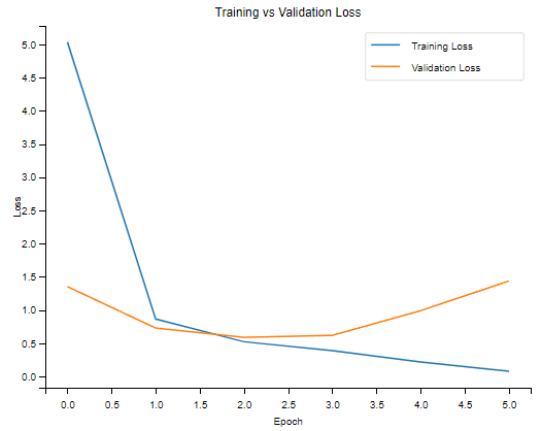

Mean Feature Importances (Instances 1-300)
Subject
Predicate
Object
Contribution to Prediction Probability

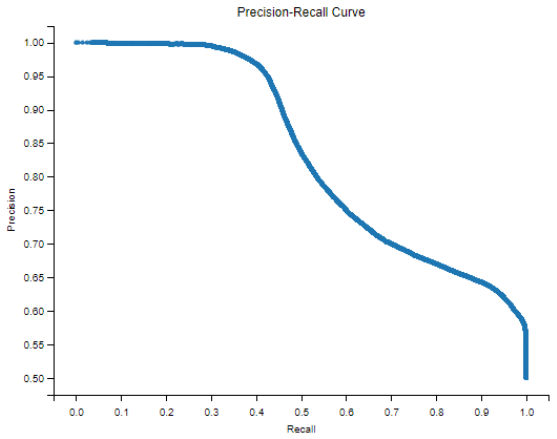

Mean Feature Importances (Instances 1000-1300)
Subject
Predicate
Object
Contribution to Prediction Probability

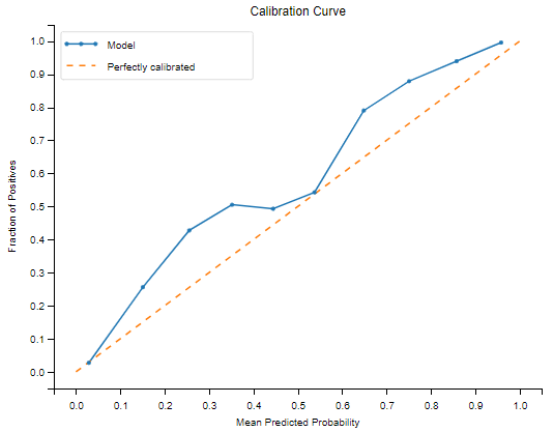


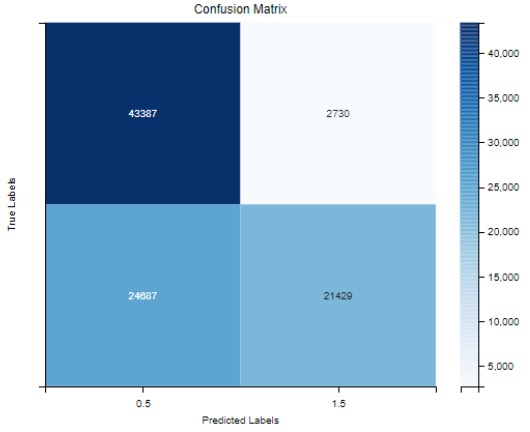
	subject	predicate	object
Triple 1	0.0119	-0.42507	0.0979
Triple 2	-0.0064	-0.41864	-0.0254
Triple 3	0.0016	0.34161	-0.0247
Triple 4	-0.0146	0.28965	-0.0199
Triple 5	0.0124	-0.4198	-0.0332

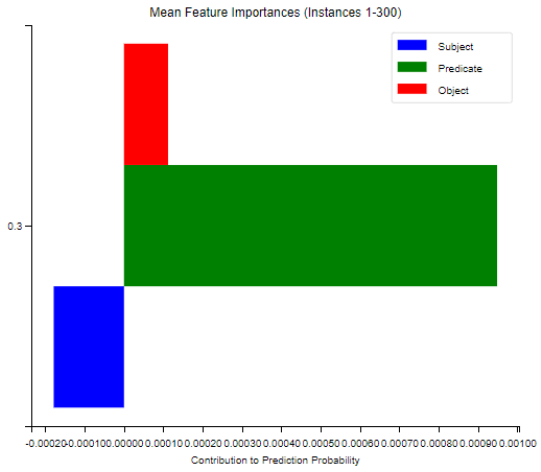
$HoIE$

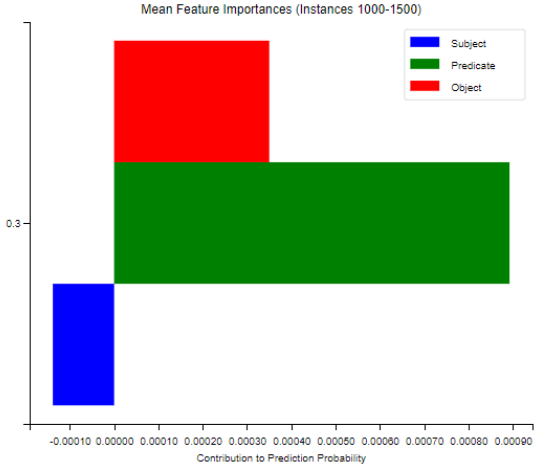


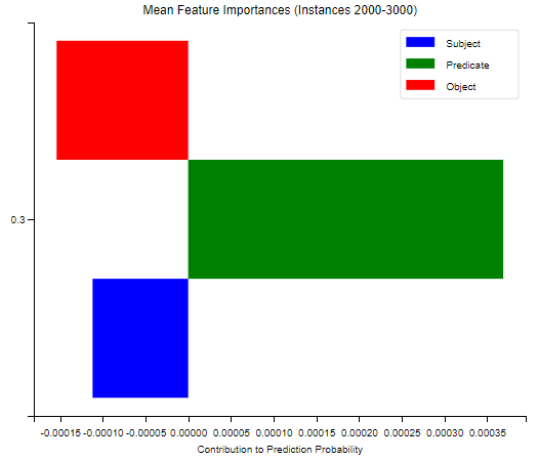






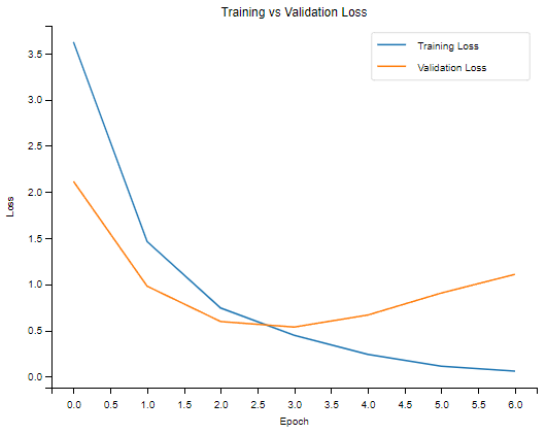


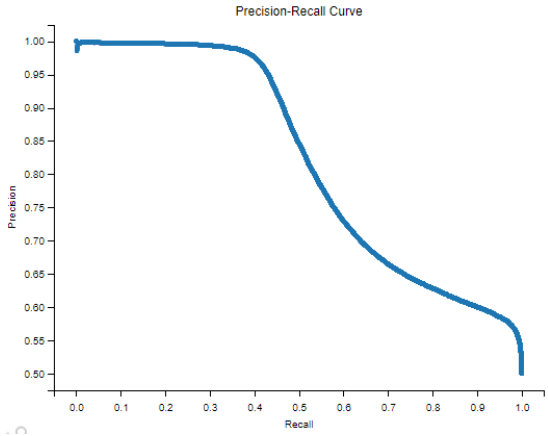


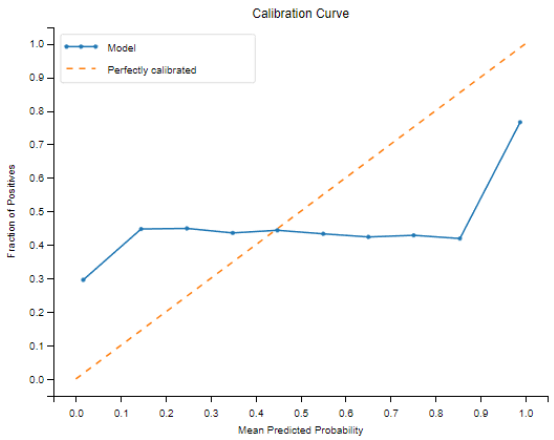


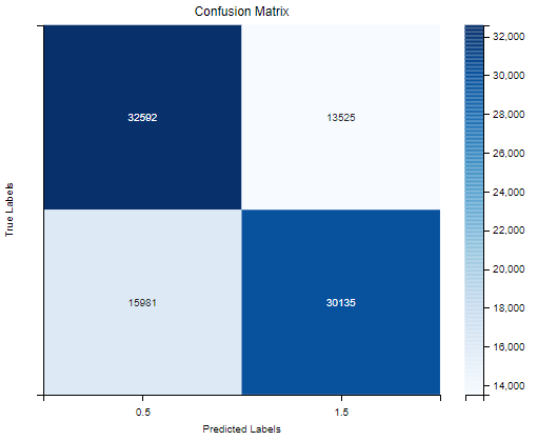
	subject	predicate	object
Triple 1	-0.0228	0.158314	0.0736
Triple 2	0.0110	-0.27657	-0.0753
Triple 3	0.0152	0.159584	0.0823
Triple 4	-0.0215	-0.28203	0.0737
Triple 5	0.0121	0.17607	-0.1159

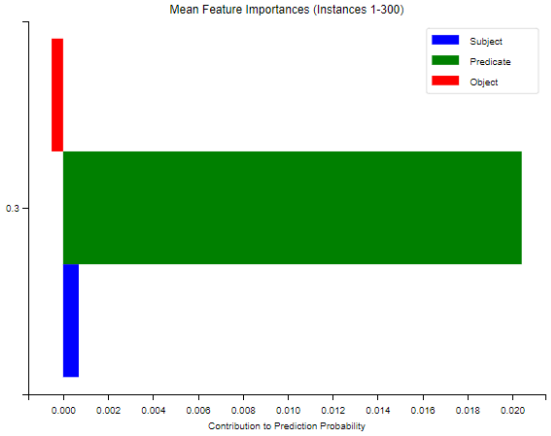
DistMult

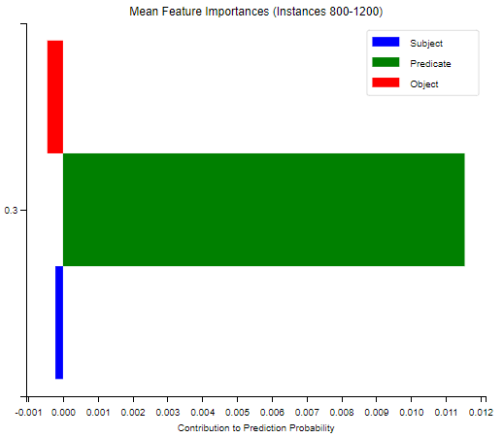


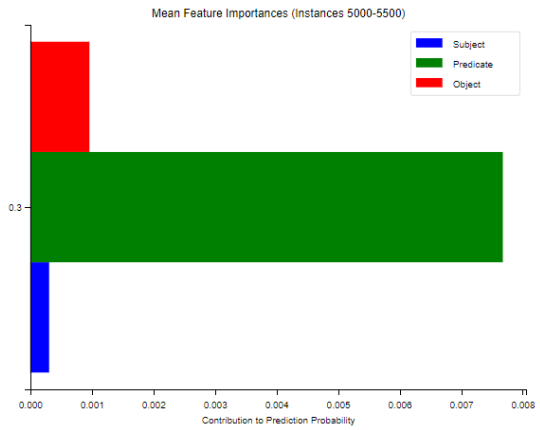












	subject	predicate	object
Triple 1	0.0758	0.36086	-0.2946
Triple 2	0.2314	-0.28855	-0.2123
Triple 3	-0.0480	-0.18071	0.2733
Triple 4	0.0141	-0.19425	0.2551
Triple 5	-0.1958	0.42322	0.0425

	C1	C2	C3	C4	C5	C6	C7	C8
$TransE$	3	1488	1480	3	0.9876	1.0000	0.0125	0.0622
$TransH$	1	0	0	0	0.5	0.5	0.5	0
$RotatE$	4	1	0	0	0.4774	0.6251	0.4411	0.0364
$NoEmbds$	7	48	0	0	0.3528	0.8432	0.0023	0.1859
$HoIE$	16	4	0	0	0.2376	0.6661	0.0008	0.1391
$DistMult$	4	405	243	4	0.3173	0.9995	0.0006	0.3777

$TransE$

Top 5 most frequent relations:
[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasURL', 1200),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasTopic', 298),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasCategoryOfGlossaryArticle', 2)
Relation with highest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasTopic', 0.9889348158300323)
Relation with lowest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasCategoryOfGlossaryArticle',
0.9263133108615875)

$RotatE$

Top 5 most frequent relations:
[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/term', 407),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/level', 377),
('http://www.w3.org/1999/02/22-rdf-syntax-ns#type', 358),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasReference', 358)]
Relation with highest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasReference',
0.47852507566606534)
Relation with lowest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/term', 0.4756814282414954)

$NoEmbds$

Top 5 most frequent relations:
[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/databasePath', 968),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasURI', 480),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/relatedTerm', 25),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/remark', 13),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/id', 12)]
Relations appearing only once:
['http://www.w3.org/2000/01/rdf-schema#label',
'https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasParagraph']
Relation with highest average probability:
('http://www.w3.org/2000/01/rdf-schema#label', 0.4436992108821869)
Relation with lowest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/id', 0.3281816483164827)

$HoIE$

Top 5 most frequent relations:
[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/term', 788),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/definition', 407),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasURI', 64),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/relatedTerm', 63),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasParagraph', 51)]
Relations appearing only once:
['http://www.w3.org/2000/01/rdf-schema#comment']
Relation with highest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/remark', 0.353840708732605)
Relation with lowest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/keyword', 0.09458667249418795)

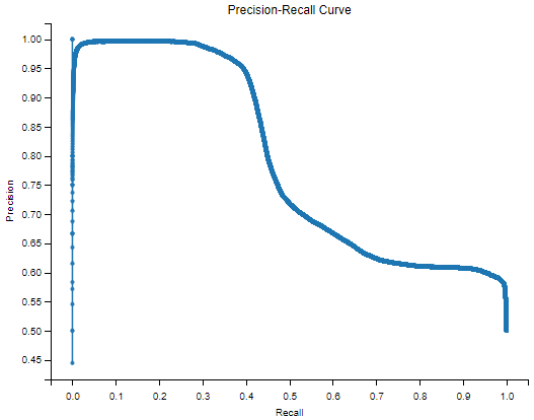
$DistMult$

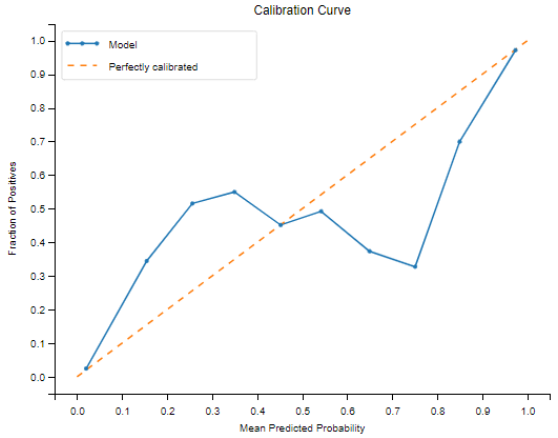
Top 5 most frequent relations:
[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasReference', 570),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/term', 520),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/level', 253),
('http://www.w3.org/1999/02/22-rdf-syntax-ns#type', 157)]
Relation with highest average probability:
('http://www.w3.org/1999/02/22-rdf-syntax-ns#type', 0.374340466957741)
Relation with lowest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/term', 0.28744966788094295)

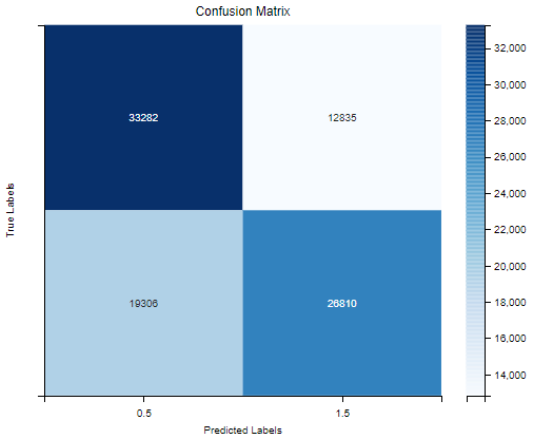
X. Group 10

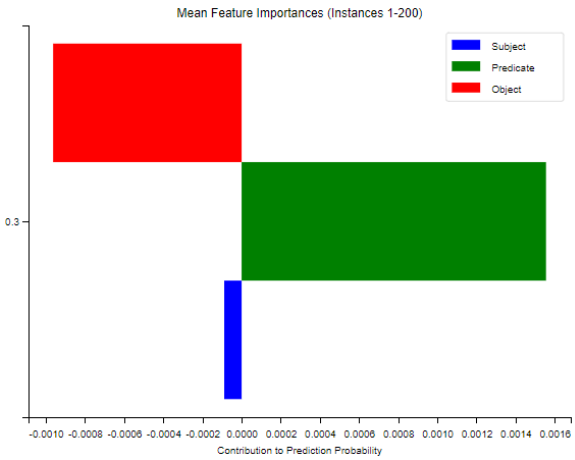
	Mean Silhouette	**Mean Davies-Bouldin**	**Mean Calinski-Harabasz**
$TransE$	0.8033	0.2534	3409.4346
$RotatE$	0.8038	0.2534	3572.9810
$NoEmbds$	0.8091	0.2453	3726.2432
$HoIE$	**0.8111**	**0.2409**	**3734.2604**
$DistMult$	0.7835	0.2864	2610.8468

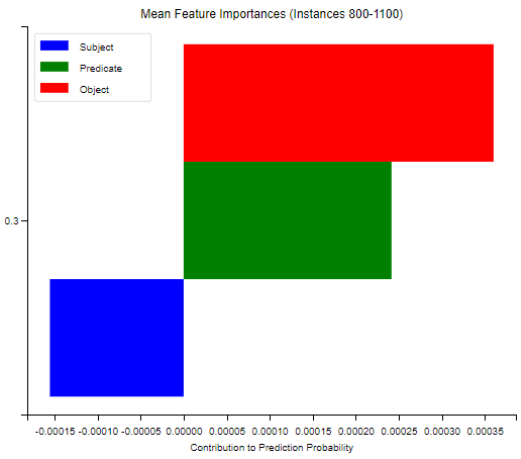
$TransE$





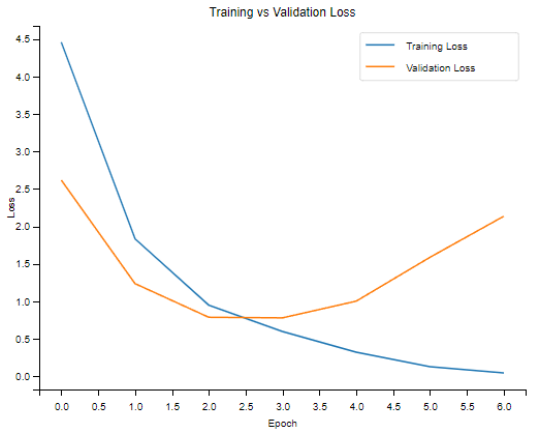


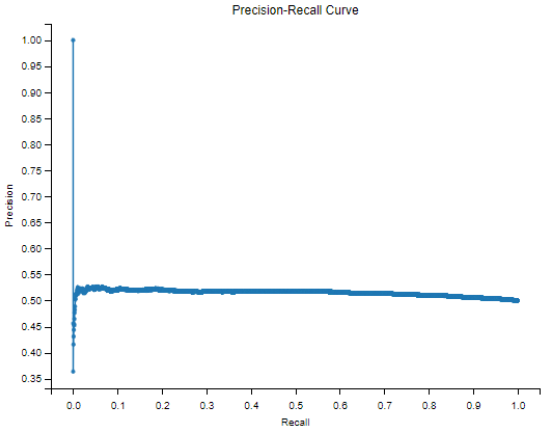


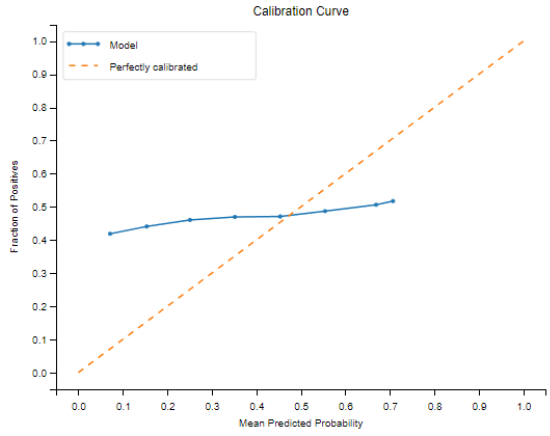


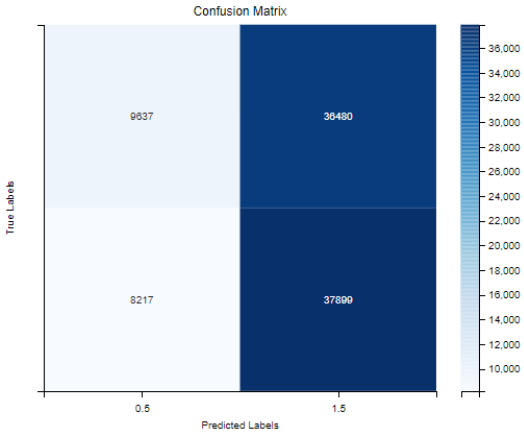
	subject	**predicate**	**object**
Triple 1	0.00238	-0.52889	0.15306
Triple 2	-0.03335	0.30291	-0.10603
Triple 3	-0.01008	0.29046	-0.08580
Triple 4	0.01318	0.30933	-0.19713
Triple 5	0.01228	-0.54333	0.13565

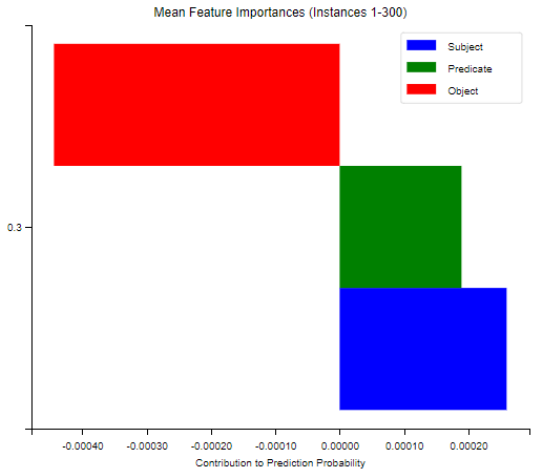
RotatE

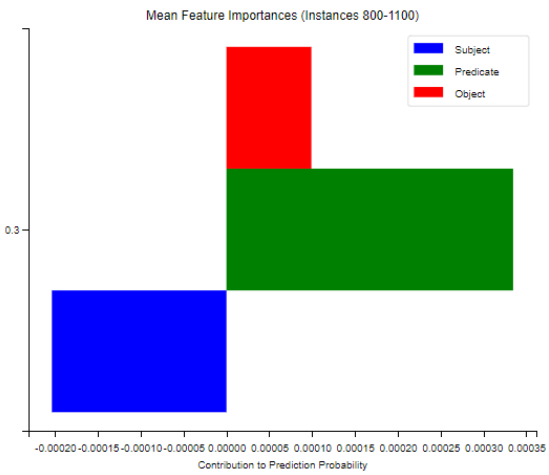

Training vs Validation Loss
Training Loss
Validation Loss
Loss
Epoch

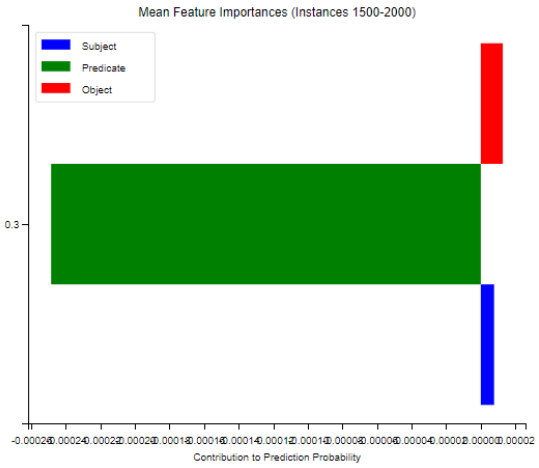

Precision-Recall Curve
Precision
Recall


Calibration Curve
Model
Perfectly calibrated
Fraction of Positives
Mean Predicted Probability


Confusion Matrix
9637
36480
8217
37899
True Labels
Predicted Labels

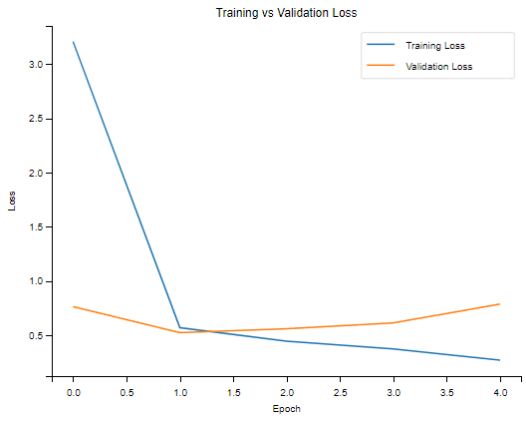

Mean Feature Importances (Instances 1-300)
Subject
Predicate
Object
Contribution to Prediction Probability

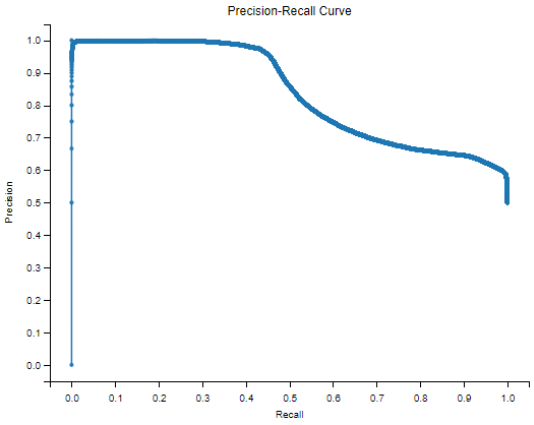

Mean Feature Importances (Instances 800-1100)
Subject
Predicate
Object
Contribution to Prediction Probability

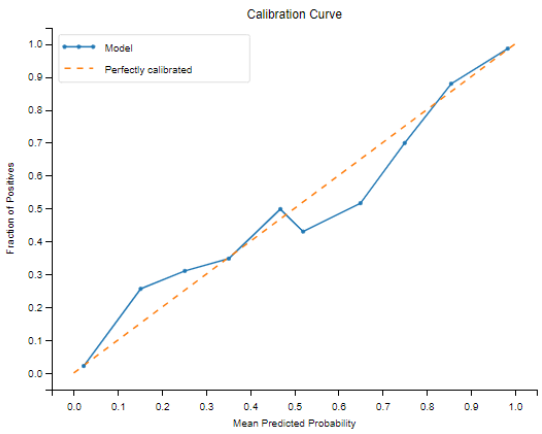


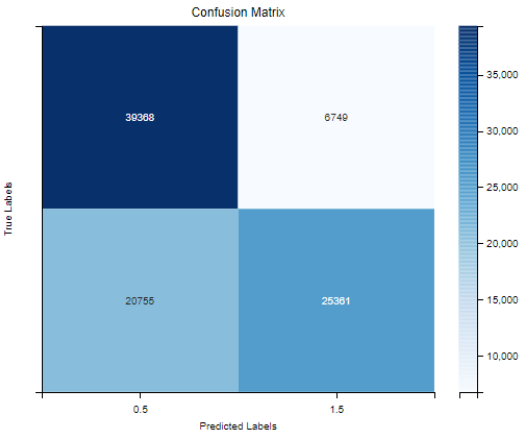
	subject	predicate	object
Triple 1	-0.0218	0.05379	-0.0389
Triple 2	0.0215	-0.03036	-0.0791
Triple 3	-0.0235	-0.05236	0.0226
Triple 4	0.0862	-0.01037	0.0438
Triple 5	-0.0222	0.05824	0.0549

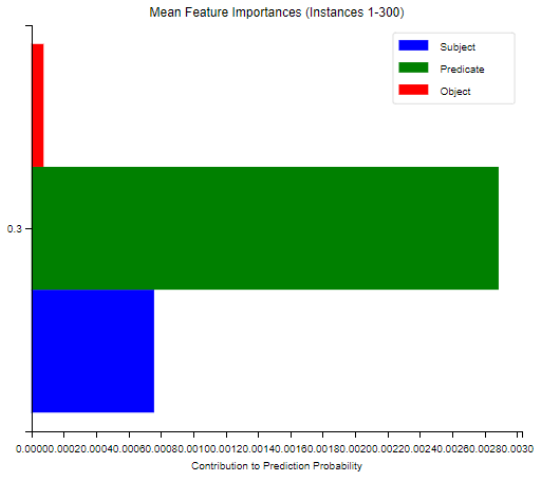
NoEmbds

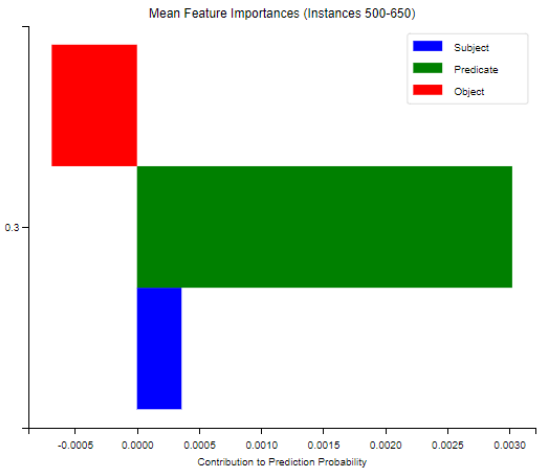


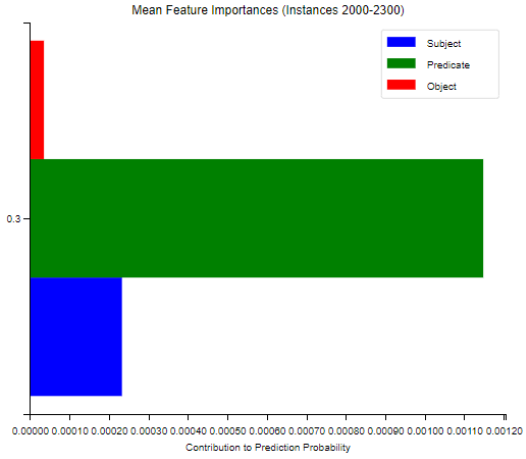






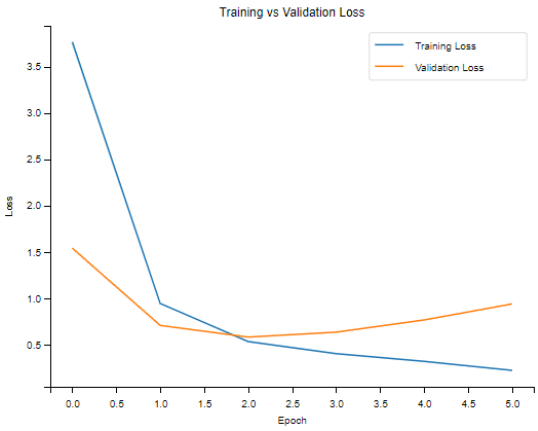


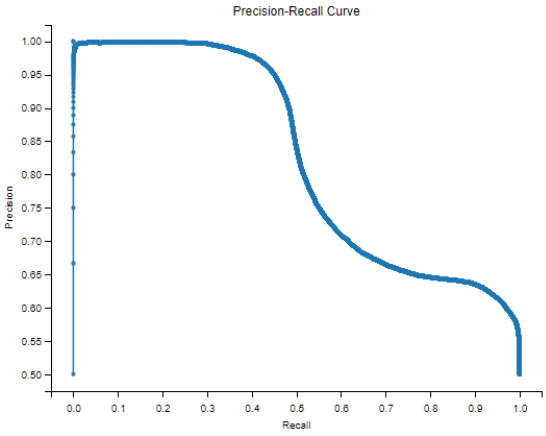


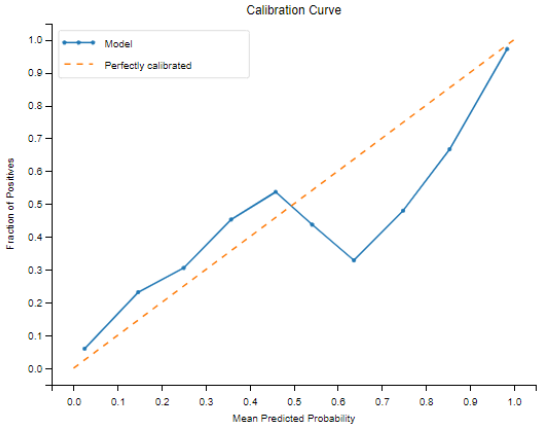


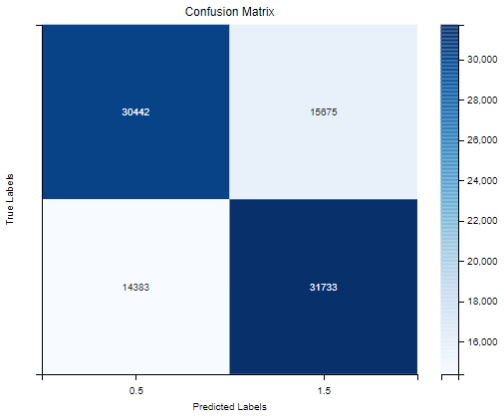
	subject	predicate	object
Triple 1	0.0109	0.10730	-0.0772
Triple 2	-0.0027	0.08269	0.1602
Triple 3	-0.0172	0.08693	-0.0772
Triple 4	-0.0141	-0.27289	-0.1055
Triple 5	0.0100	0.07761	0.1677

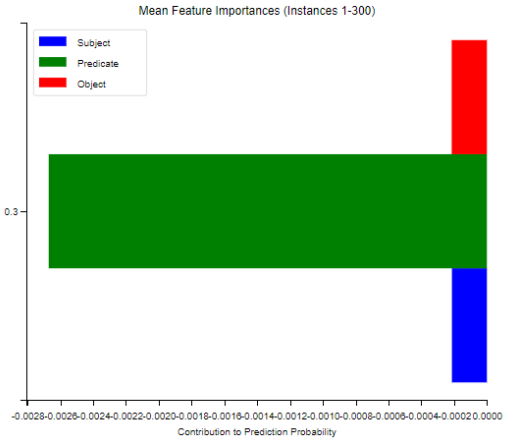
$HoIE$

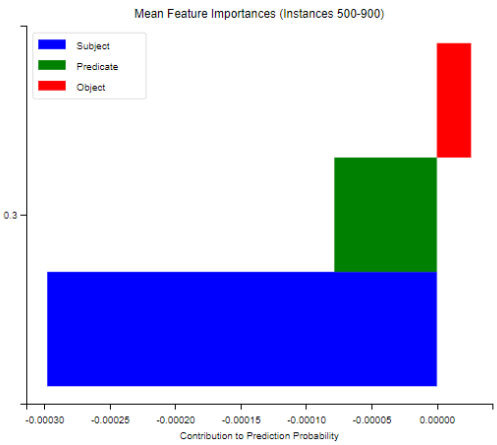


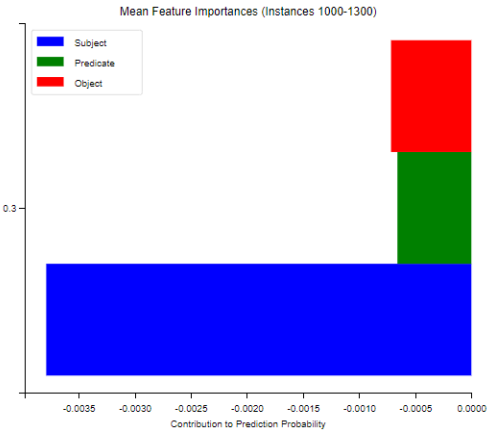








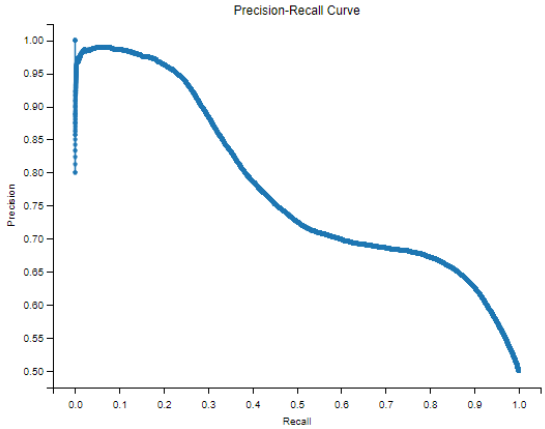


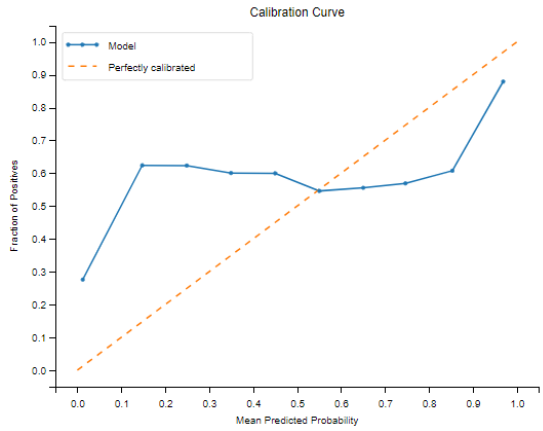


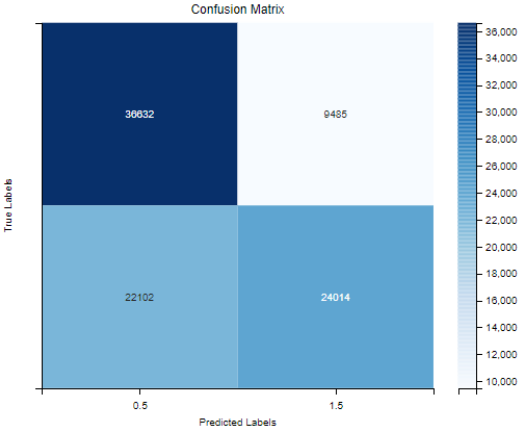
	subject	predicate	object
Triple 1	-0.01022	-0.0860	-0.11667
Triple 2	0.00281	0.30978	0.25959
Triple 3	-0.0070	0.33680	-0.13137
Triple 4	-0.00131	-0.09063	-0.19342
Triple 5	0.00742	-0.41156	0.26631

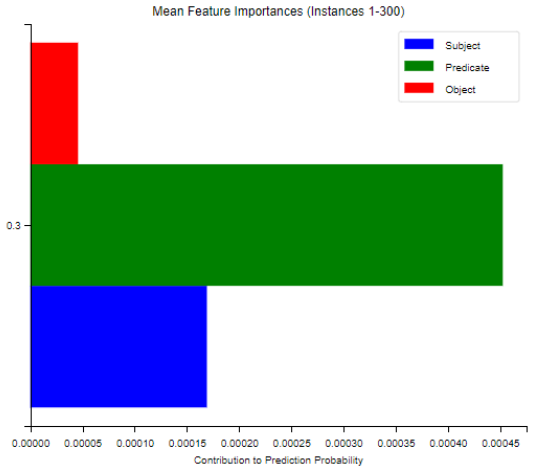
DistMult

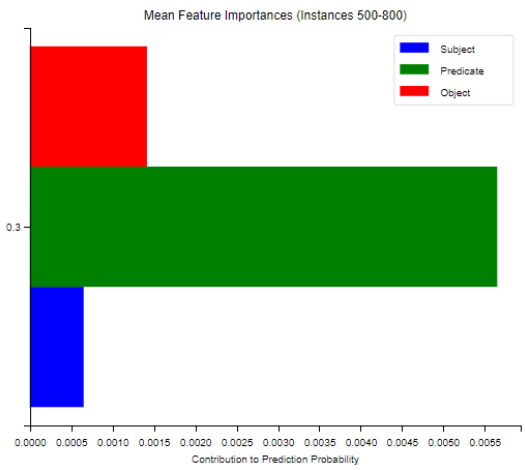

Training vs Validation Loss
Training Loss
Validation Loss
Loss
Epoch

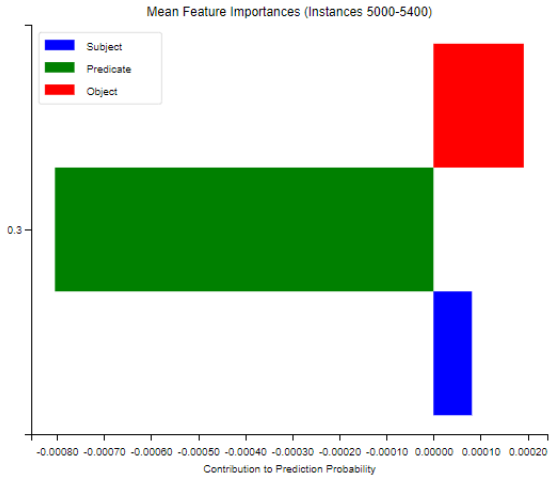

Precision-Recall Curve
Precision
Recall


Calibration Curve
Model
Perfectly calibrated
Fraction of Positives
Mean Predicted Probability


Confusion Matrix
36632
9485
22102
24014
True Labels
Predicted Labels


Mean Feature Importances (Instances 1-300)
Subject
Predicate
Object
Contribution to Prediction Probability


Mean Feature Importances (Instances 500-800)
Subject
Predicate
Object
Contribution to Prediction Probability



	subject	predicate	object
Triple 1	-0.0303	-0.27359	0.0807
Triple 2	0.0450	0.42939	0.0058
Triple 3	-0.0086	-0.27712	0.0708
Triple 4	-0.0125	0.02860	-0.0913
Triple 5	0.0023	-0.06758	-0.0674

	C1	C2	C3	C4	C5	C6	C7	C8
$TransE$	4	200	0	0	0.3217	0.8646	0.0008	0.2288
$RotatE$	4	1046	0	0	0.6000	0.7093	0.0311	0.1664
$NoEmbds$	5	70	9	3	0.3306	0.9496	0.0007	0.2144
$HolE$	3	241	15	1	0.3399	0.9980	0.0001	0.2610
$DistMult$	3	250	72	2	0.2341	0.9949	0.0001	0.3235

$TransE$

Top 5 most frequent relations:
[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/databasePath', 1193),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/id', 179),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/relatedTerm', 98),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasURI', 30)]
Relation with highest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/relatedTerm', 0.3323723630794343)
Relation with lowest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/id', 0.3133362166989366)

$RotatE$

Top 5 most frequent relations:
[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/level', 421),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasReference', 399),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/term', 359),
('http://www.w3.org/1999/02/22-rdf-syntax-ns#type', 321)]
Relation with highest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasReference', 0.603123583039619)
Relation with lowest average probability:
('http://www.w3.org/1999/02/22-rdf-syntax-ns#type', 0.5962778416250446)

$NoEmbds$

Top 5 most frequent relations:
[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasParagraph', 888),
('http://www.w3.org/2002/07/owl#unionOf', 232),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/relatedTerm', 204),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/id', 165),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/title', 11)]
Relation with highest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/title', 0.3658370489508591)
Relation with lowest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/id', 0.31402607393013593)

$HoIE$

Top 5 most frequent relations:
[('http://www.w3.org/1999/02/22-rdf-syntax-ns#type', 1180),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/level', 306),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/term', 6)]
Relation with highest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/level', 0.35716361697782034)
Relation with lowest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/term', 0.3028070529301961)

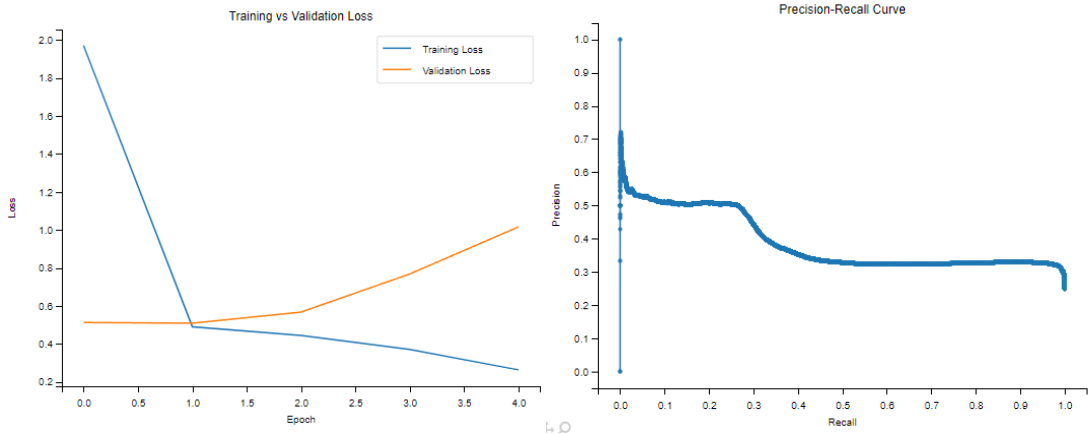
$DistMult$

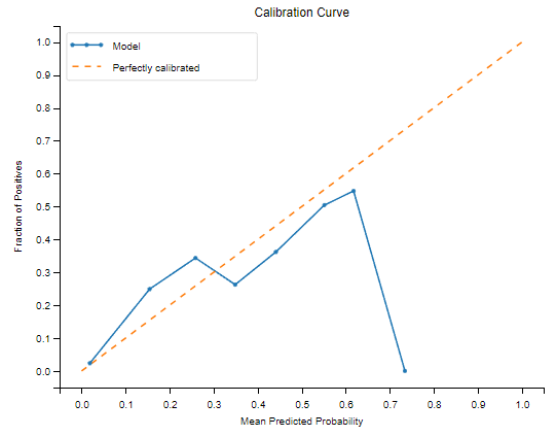
Top 5 most frequent relations:
[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/level', 1196),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasReference', 35),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/term', 1)]
Relations appearing only once:
['https://ec.europa.eu/eurostat/NLP4StatRef/ontology/term']
Relation with highest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/level', 0.23641097045408993)
Relation with lowest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/term', 0.12646132707595825)

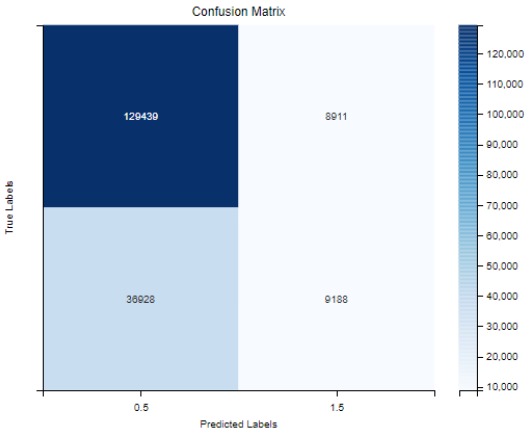
XI. Experiment 1

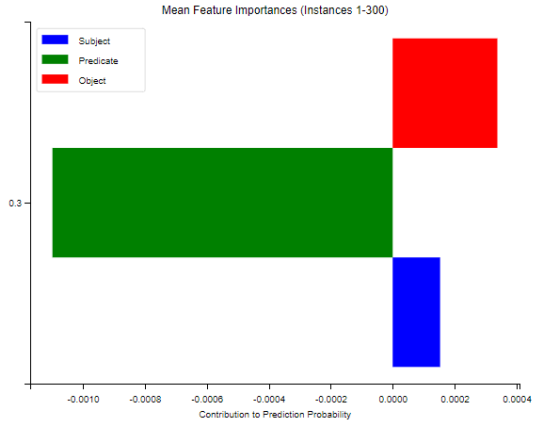
	Mean Silhouette	Mean Davies-Bouldin	Mean Calinski-Harabasz
	0.7622	0.2842	3048.9623

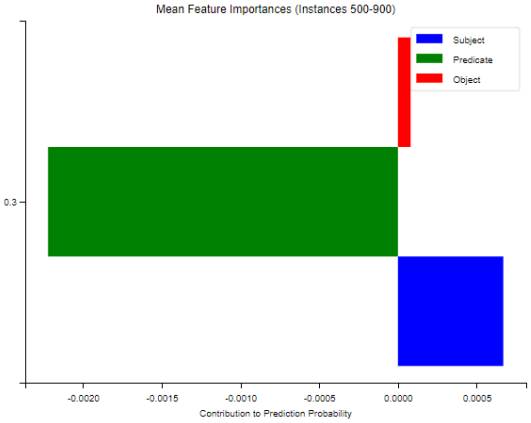
C1	C2	C3	C4	C5	C6	C7	C8
16	0	0	0	0.1436	0.4584	0.0001	0.4584

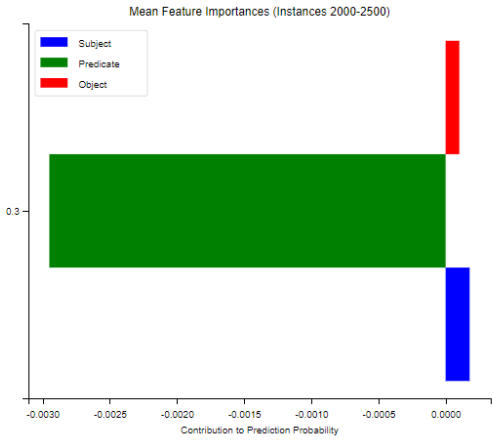












	subject	predicate	object
Triple 1	0.0032	-0.22903	-0.0183
Triple 2	0.0146	0.01207	-0.0294
Triple 3	0.0148	0.14214	0.0480
Triple 4	-0.0160	0.00884	0.0209
Triple 5	-0.0146	0.13791	-0.0332

Top 5 most frequent relations:

[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/term', 359),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/definition', 286),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasURL', 188),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasCode', 77),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/title', 53)]

Relations appearing only once:

['https://ec.europa.eu/eurostat/NLP4StatRef/ontology/sourcePublication']

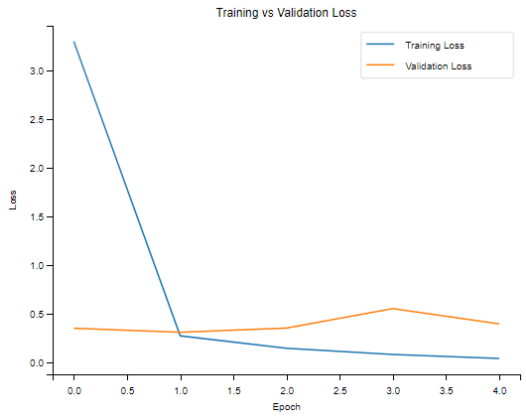
Relation with highest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/sourcePublication',
0.36843347549438477)

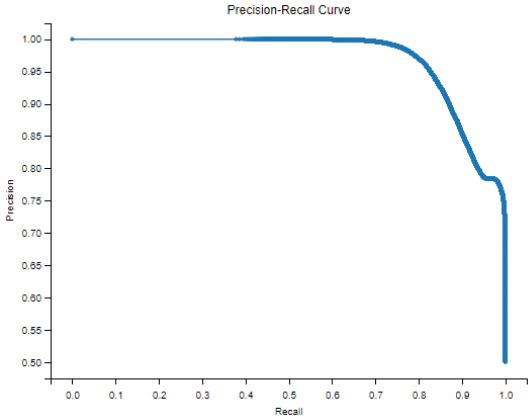
Relation with lowest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/dateUpdated',
0.032192379236221313)

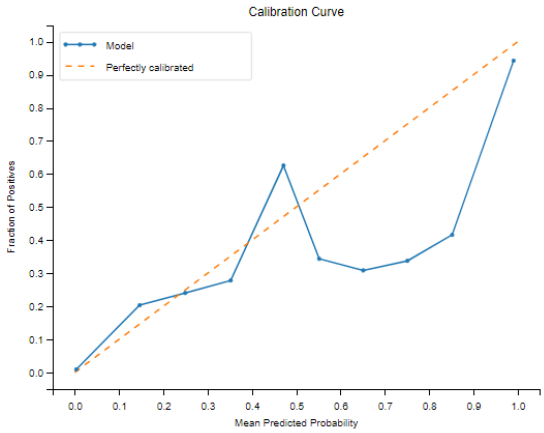
XII. Experiment 2

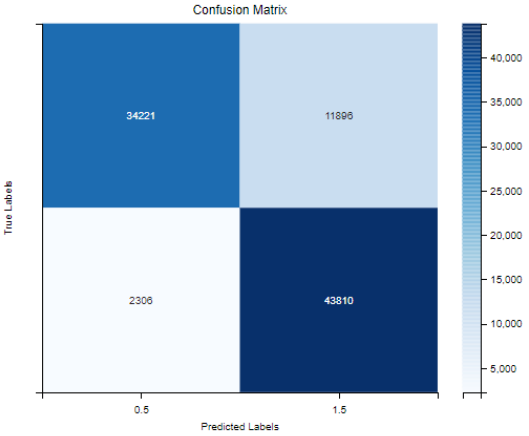
	Mean Silhouette	Mean Davies-Bouldin	Mean Calinski-Harabasz
	0.7266	0.2940	2156.5329

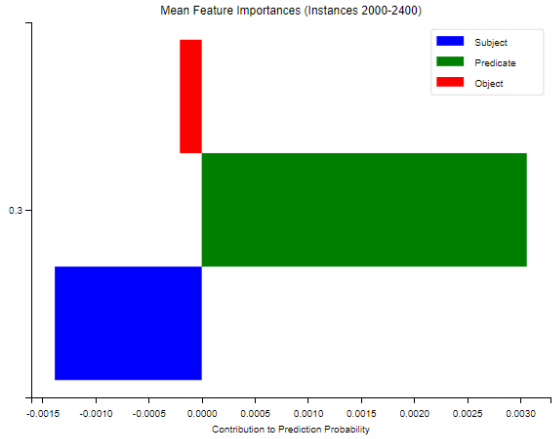
	C1	C2	C3	C4	C5	C6	C7	C8
	25	510	331	21	0.5692	1.0000	0.0000	0.3720

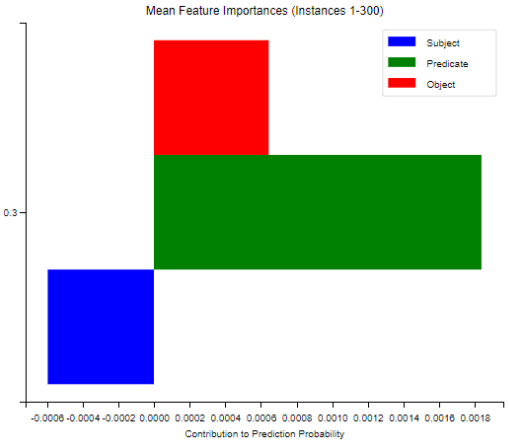


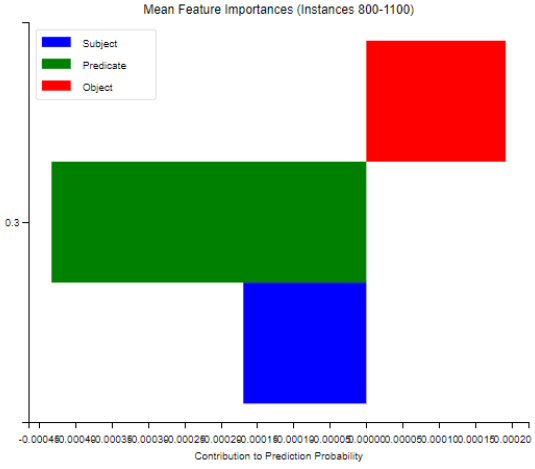












	subject	predicate	object
Triple 1	-0.0522	0.09714	-0.0709
Triple 2	0.0608	0.34187	0.1169
Triple 3	0.0182	-0.64982	-0.1923
Triple 4	0.0008	0.33244	0.2700
Triple 5	-0.0339	0.12016	-0.1793

Top 5 most frequent relations:

[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/definition', 283),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/term', 199),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasCode', 157),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/content', 72),
('http://www.w3.org/2000/01/rdf-schema#label', 64)]

Relations appearing only once:

['https://ec.europa.eu/eurostat/NLP4StatRef/ontology/context',
'https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasTopic']

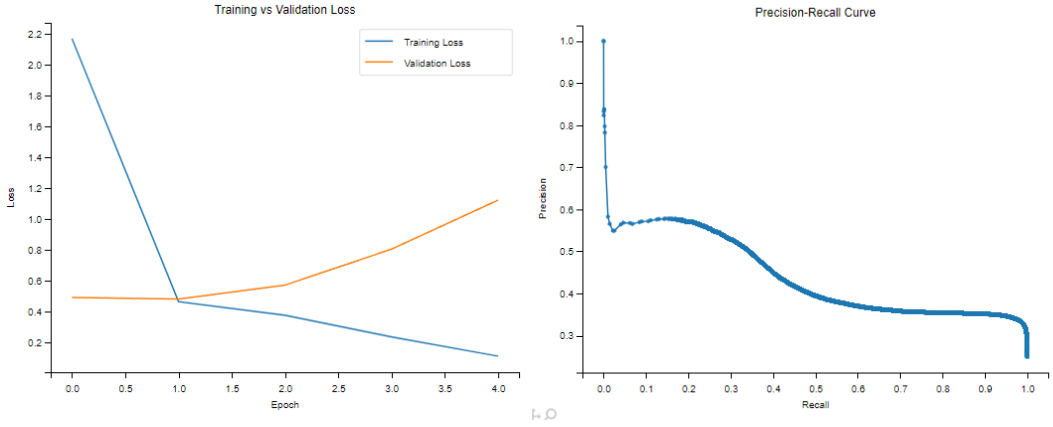
Relation with highest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasCategoryOfGlossaryArticle',
0.7590611591935158)

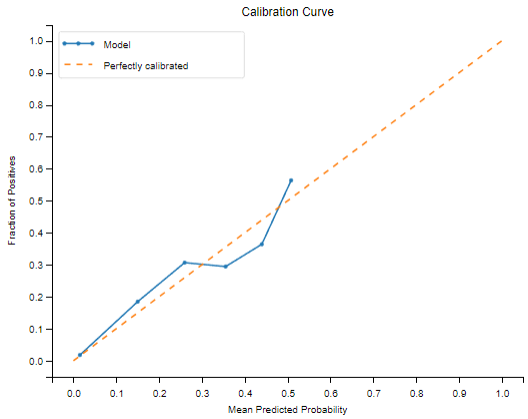
Relation with lowest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasURI', 0.12462425937216419)

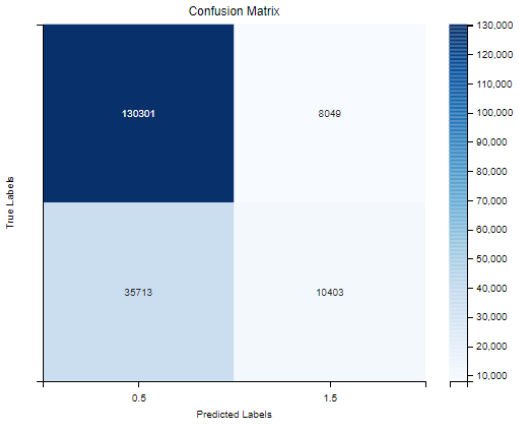
XIII. Experiment 3

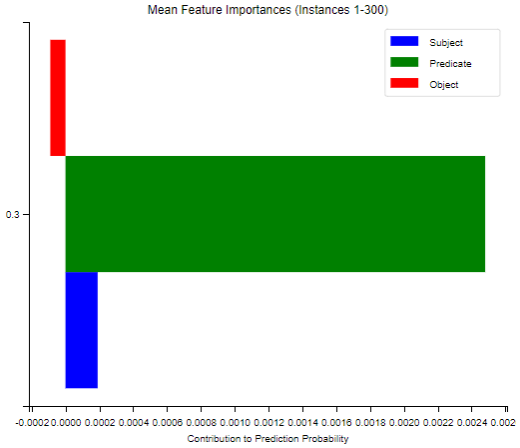
	Mean Silhouette	Mean Davies-Bouldin	Mean Calinski-Harabasz
	0.7541	0.2978	2558.1597

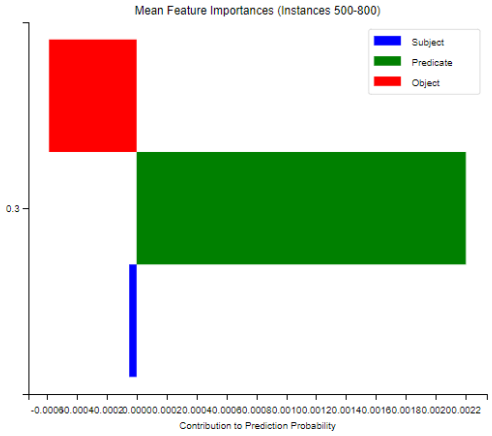
	C1	C2	C3	C4	C5	C6	C7	C8
	16	0	0	0	0.1592	0.4830	0.0001	0.1557

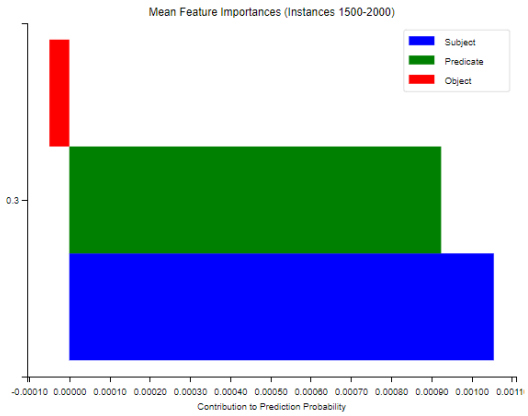












	subject	predicate	object
Triple 1	-0.0173	-0.1465	-0.0018
Triple 2	-0.0095	-0.14326	-0.0079
Triple 3	0.0343	0.33986	0.0005
Triple 4	-0.0276	-0.02431	-0.0094
Triple 5	0.0328	-0.10442	0.0126

Top 5 most frequent relations:

[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasURL', 471),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/term', 294),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasURI', 107),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasReference', 41),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/title', 38)]

Relations appearing only once:

['http://www.w3.org/2000/01/rdf-schema#label',
'https://ec.europa.eu/eurostat/NLP4StatRef/ontology/fileLink',
'https://ec.europa.eu/eurostat/NLP4StatRef/ontology/dateUpdated']

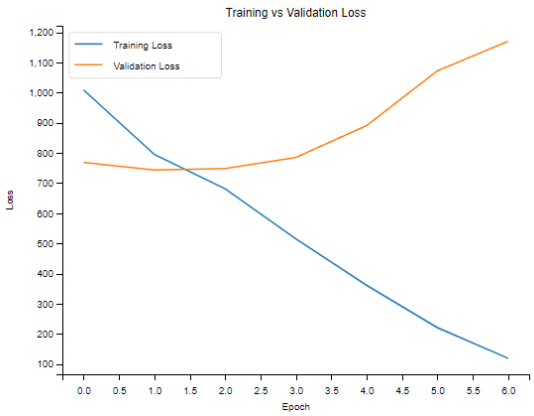
Relation with highest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/relatedTerm',
0.25532177307953435)

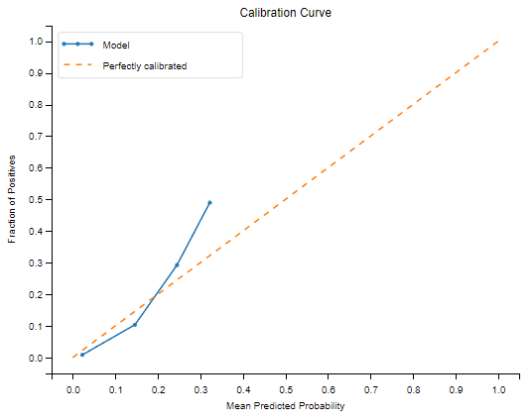
Relation with lowest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/fileLink', 0.005369111895561218)

XIV. Experiment 4

	Mean Silhouette	Mean Davies-Bouldin	Mean Calinski-Harabasz
	0.8118	0.2401	3526.3153

	C1	C2	C3	C4	C5	C6	C7	C8
	2	0	0	0	0.0961	0.2954	0.0001	0.0937





Top 5 most frequent relations:

[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/level', 1417),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasReference', 25)]

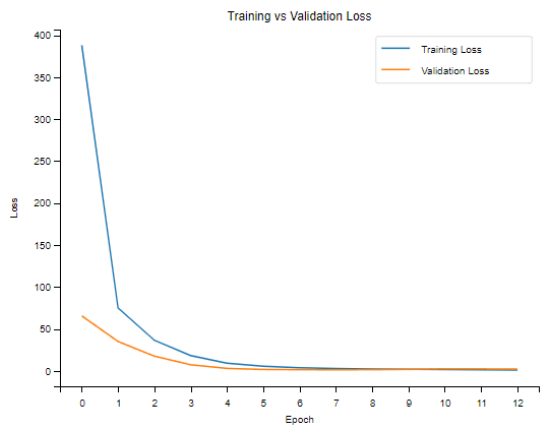
Relation with highest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/level', 0.09643133350456623)

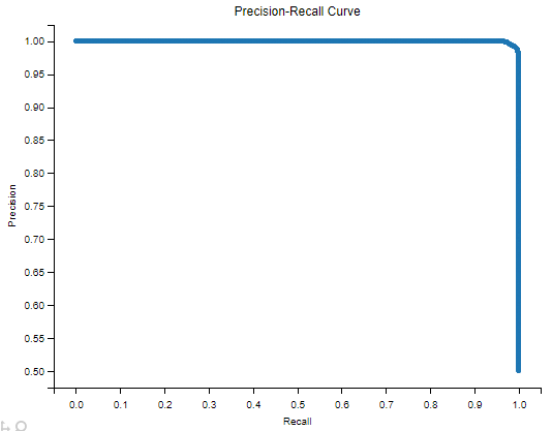
Relation with lowest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasReference',
0.07899965308519313)

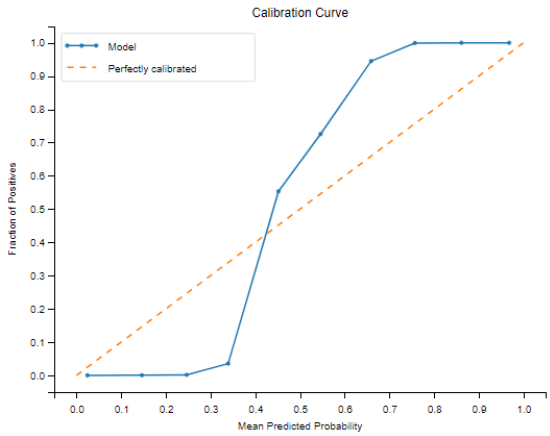
XV. Experiment 5

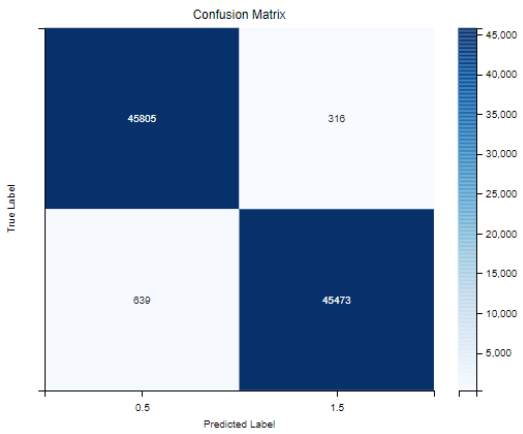
	Mean Silhouette	Mean Davies-Bouldin	Mean Calinski-Harabasz
	0.7967	0.2659	4258.8930

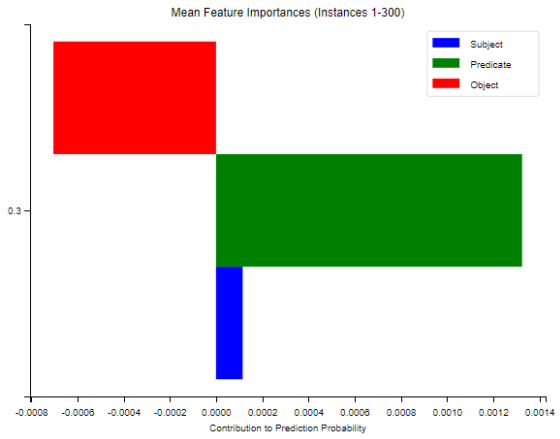
C1	C2	C3	C4	C5	C6	C7	C8
42	1409	436	40	0.8345	0.9965	0.1408	0.1238

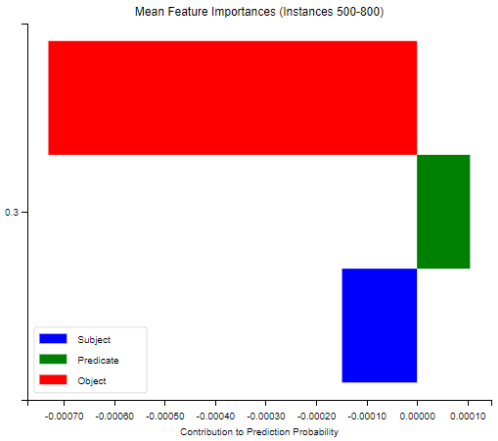


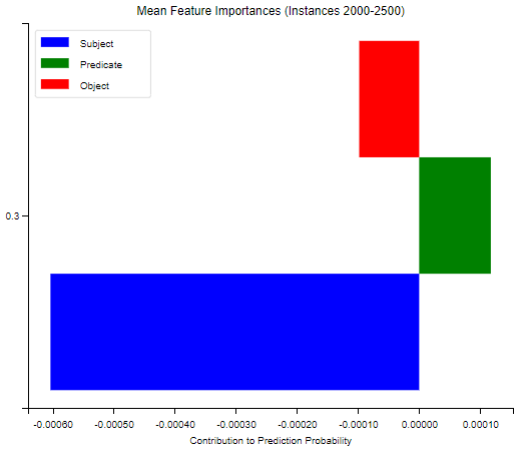












	subject	predicate	object
Triple 1	0.0953	0.08482	0.0807
Triple 2	-0.1829	0.09034	-0.0509
Triple 3	0.0683	-0.12324	0.0918
Triple 4	0.0872	0.0867	-0.0655
Triple 5	-0.0222	-0.07559	-0.0103

Top 5 most frequent relations:

[('http://www.w3.org/2000/01/rdf-schema#comment', 134),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/sourcePublication', 108),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/remark', 81),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasCode', 79),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasFrequentTerm', 74)]

Relations appearing only once:

['http://www.w3.org/2002/07/owl#backwardCompatibleWith']

Relation with highest average probability:

('http://www.w3.org/2002/07/owl#equivalentClass', 0.9076344115393502)

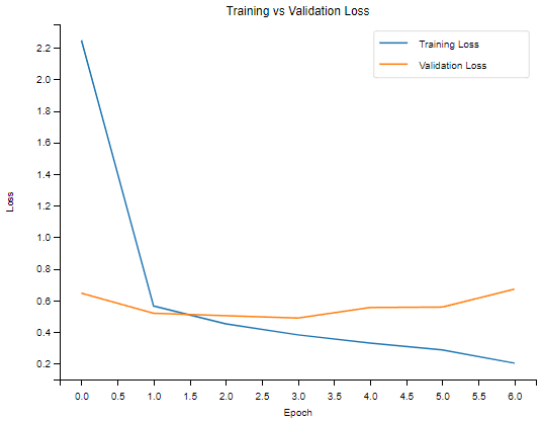
Relation with lowest average probability:

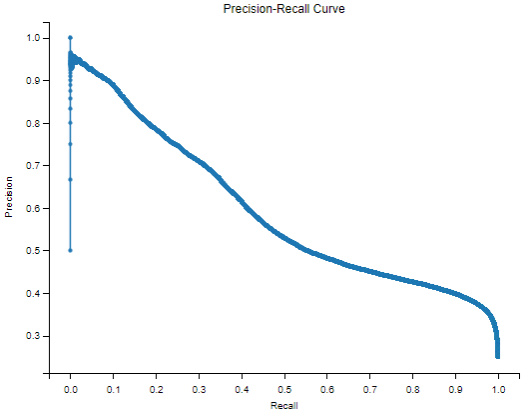
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/level', 0.7821886017918587)

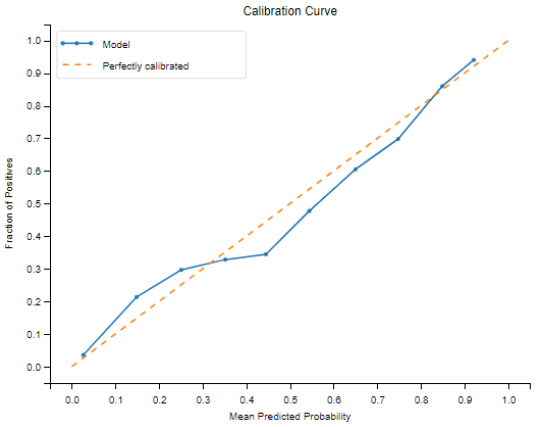
XVI. Experiment 6

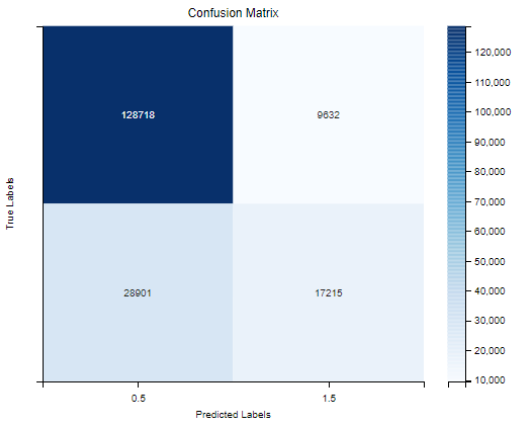
	Mean Silhouette	Mean Davies-Bouldin	Mean Calinski-Harabasz
	0.8027	0.2564	3468.3677

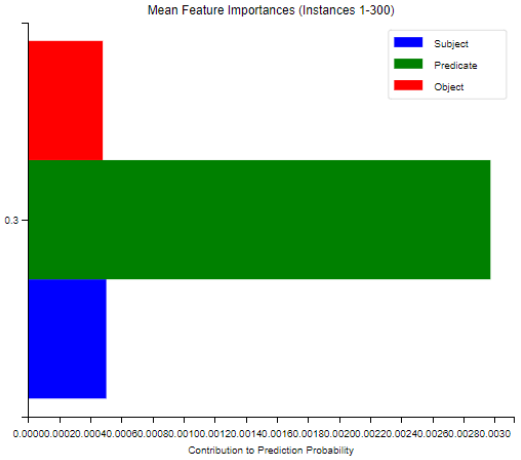
C1	C2	C3	C4	C5	C6	C7	C8
22	2	0	0	0.0760	0.6515	0.0001	0.1211

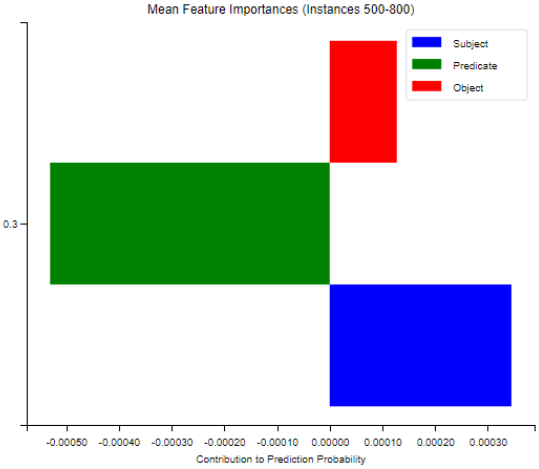


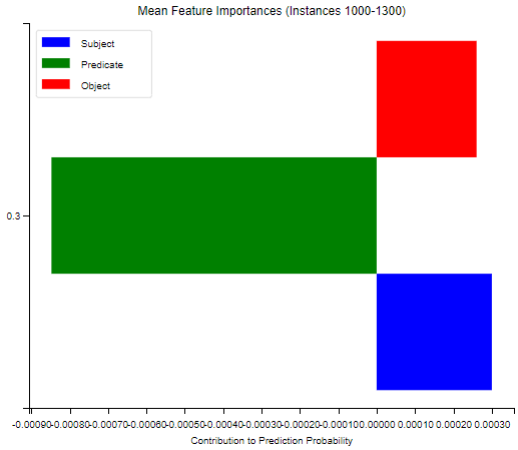












	subject	predicate	object
Triple 1	-0.0543	-0.0580	0.0001
Triple 2	0.1938	0.13990	0.0216
Triple 3	-0.0629	0.11025	0.0007
Triple 4	-0.0295	-0.04595	-0.0220
Triple 5	-0.0445	-0.05756	0.0026

Top 5 most frequent relations:

[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/remark', 766),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/definition', 208),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasCode', 83),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/databasePath', 55),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasURI', 46)]

Relations appearing only once:

['https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasCategoryOfStatisticExplainedArticle', 'https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasOECDTheme', 'https://ec.europa.eu/eurostat/NLP4StatRef/ontology/dataSource']

Relation with highest average probability:

('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasCategoryOfStatisticExplainedArticle', 0.4231768250465393)

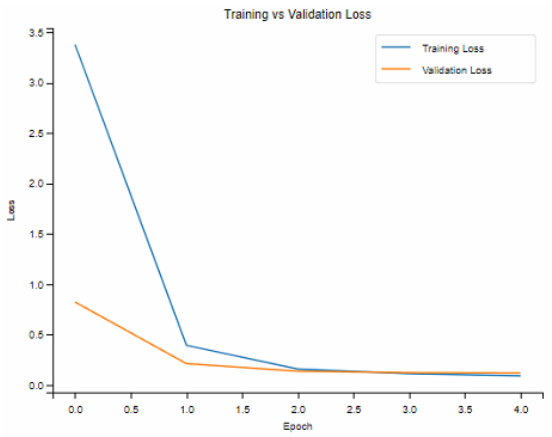
Relation with lowest average probability:

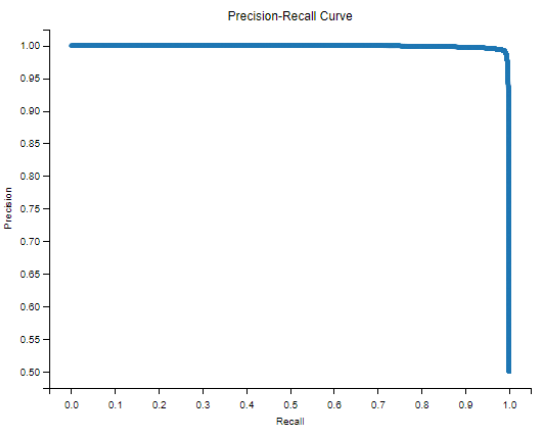
('http://www.w3.org/2000/01/rdf-schema#subClassOf', 0.007155439350754023)

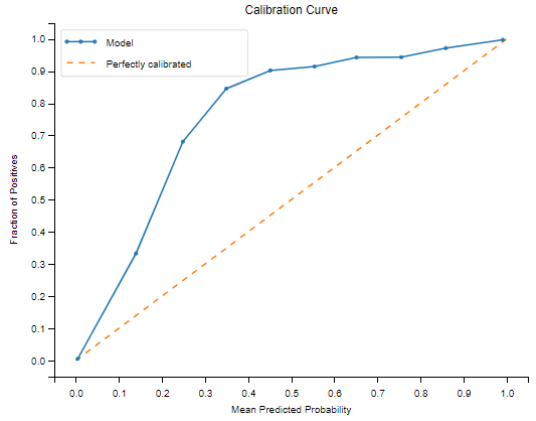
XVII. Experiment 7

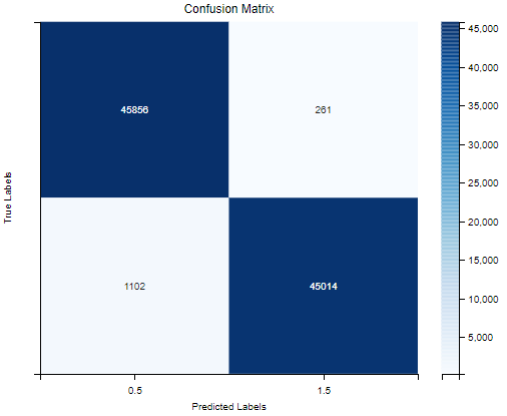
	Mean Silhouette	Mean Davies-Bouldin	Mean Calinski-Harabasz
	0.8069	0.2479	3895.3840

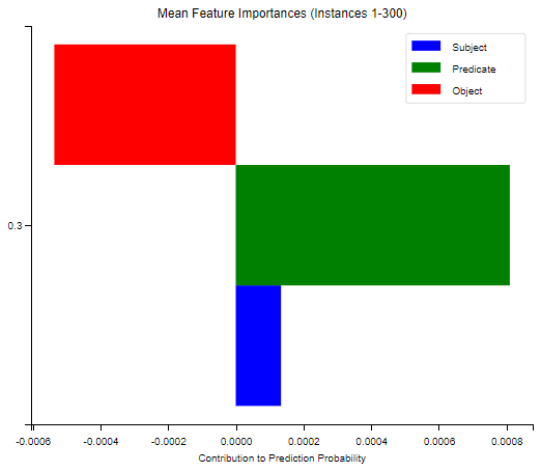
	C1	C2	C3	C4	C5	C6	C7	C8
	11	937	659	11	0.6658	1.0000	0.0039	0.3460

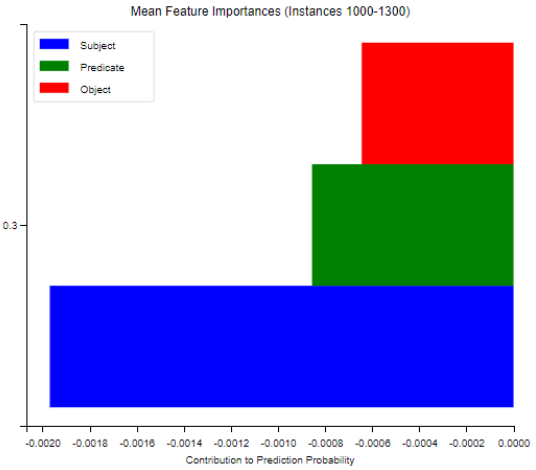


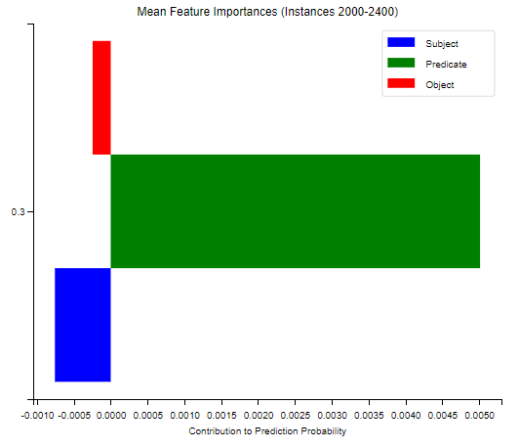












	subject	predicate	object
Triple 1	0.0824	-0.4168	0.2396
Triple 2	-0.0829	-0.11272	-0.2867
Triple 3	-0.0530	0.39615	-0.1675
Triple 4	0.0568	-0.41926	0.1248
Triple 5	0.0186	0.3313	0.2138

Top 5 most frequent relations:

[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasCode', 692),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/databasePath', 197),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasURI', 190),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/term', 125),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/definition', 102)]

Relations appearing only once:

['https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasFrequentTerm']

Relation with highest average probability:

('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasFrequentTerm',
0.947577178478241)

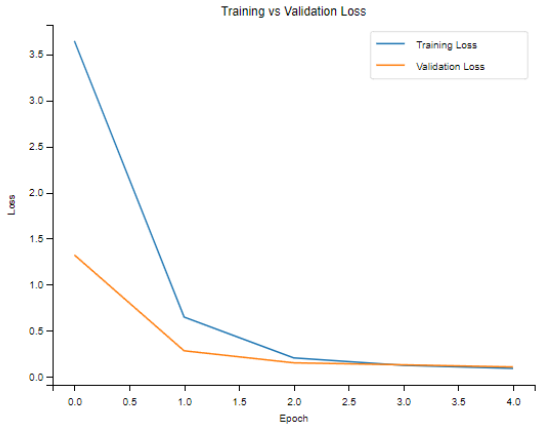
Relation with lowest average probability:

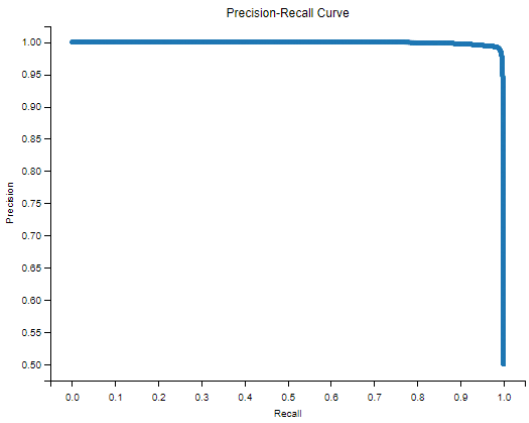
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/definition', 0.6081479691498566)

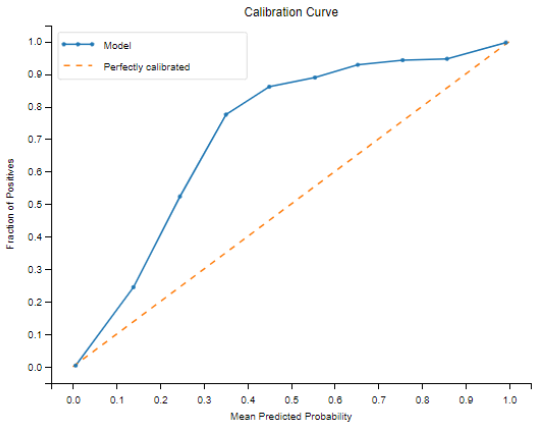
XVIII. Experiment 8

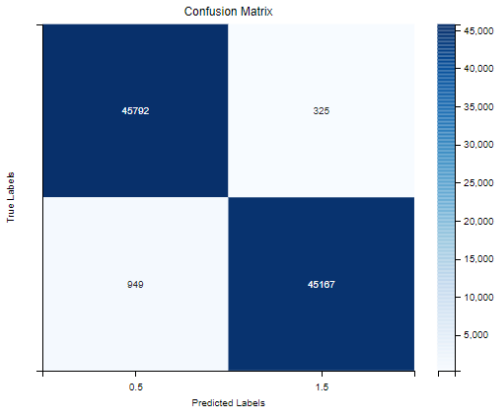
	Mean Silhouette	Mean Davies-Bouldin	Mean Calinski-Harabasz
	0.7985	0.2603	3251.2921

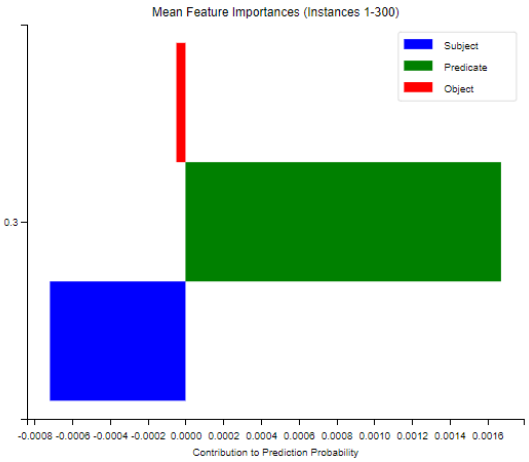
C1	C2	C3	C4	C5	C6	C7	C8
5	1104	842	5	0.7571	1.0000	0.0017	0.3068

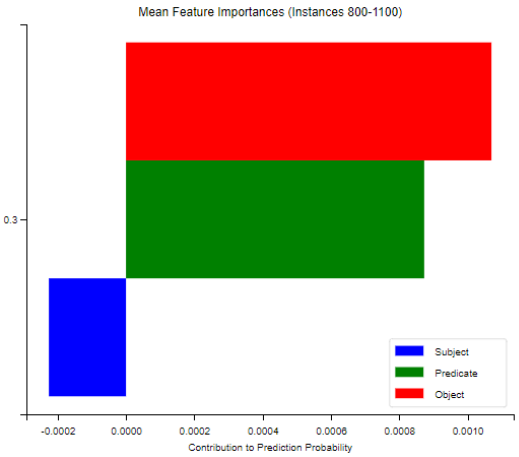


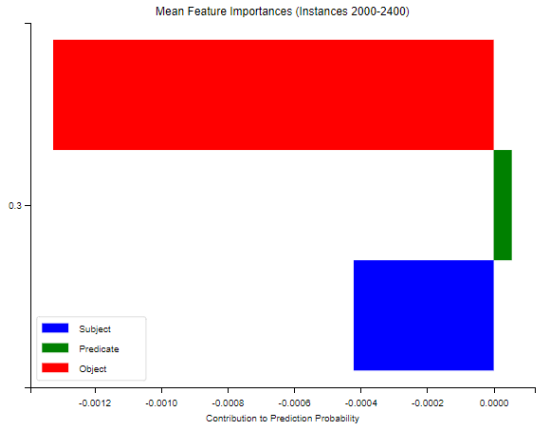












	subject	predicate	object
Triple 1	-0.1499	0.04585	0.0012
Triple 2	-0.1542	-0.21939	-0.0807
Triple 3	-0.2188	-0.21485	-0.0145
Triple 4	0.0407	0.16397	-0.3635
Triple 5	0.3796	0.0969	0.3798

Top 5 most frequent relations:

[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasCode', 1369),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasURI', 122),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/fileLink', 3),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/relatedTerm', 3),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/content', 3)]

Relation with highest average probability:

('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/relatedTerm', 0.9976281921068827)

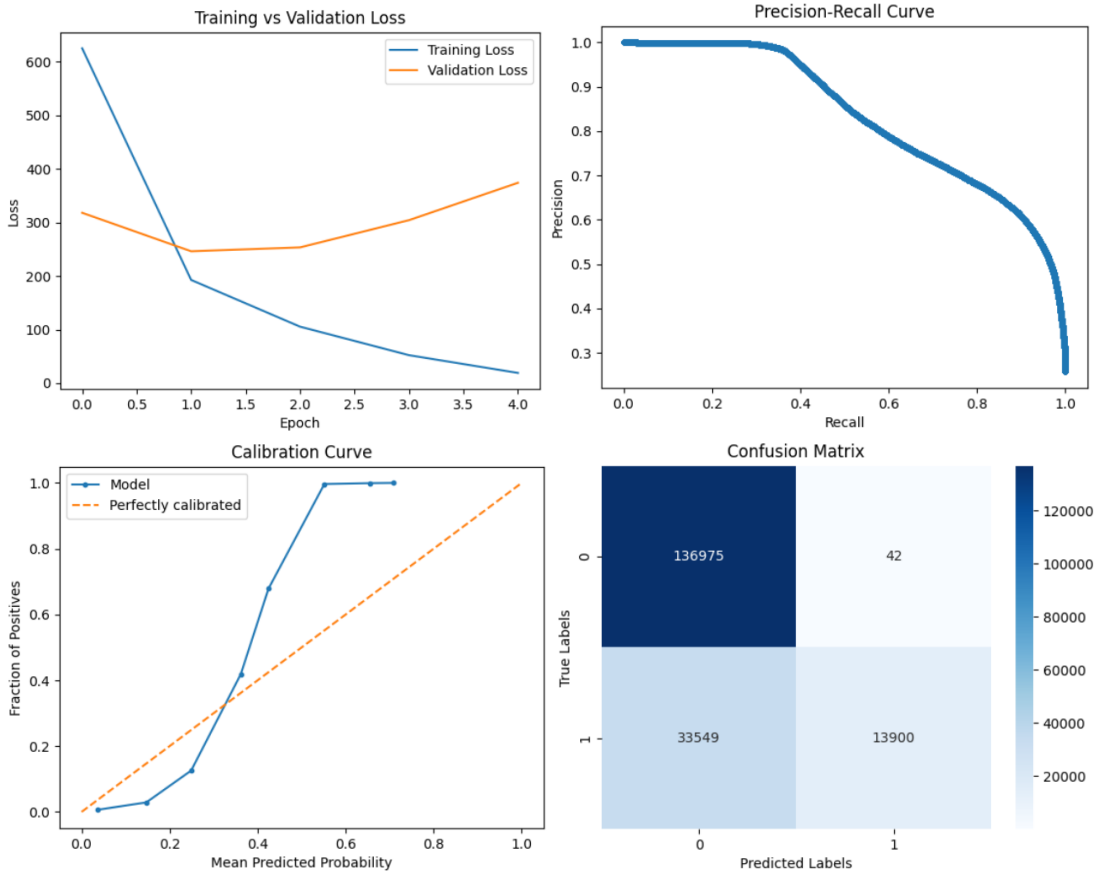
Relation with lowest average probability:

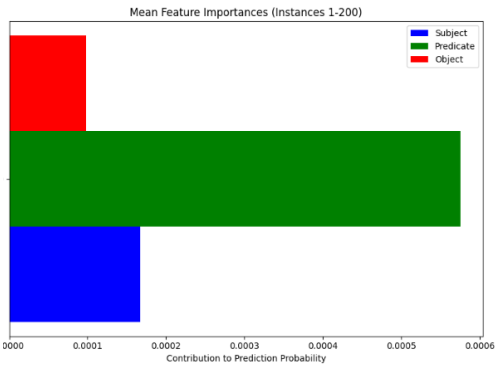
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/content', 0.5336959660053253)

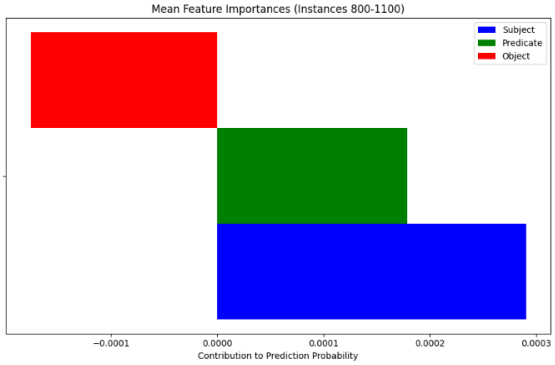
XIX. Experiment 9

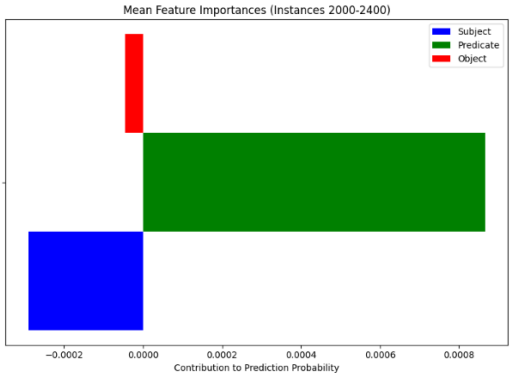
	Mean Silhouette	Mean Davies-Bouldin	Mean Calinski-Harabasz
	0.8068	0.2487	3582.4176

	C1	C2	C3	C4	C5	C6	C7	C8
	4	0	0	0	0.2678	0.4549	0.0050	0.1117









	subject	predicate	object
Triple 1	-0.1092	-0.13983	-0.1011
Triple 2	-0.0008	0.02448	0.0370
Triple 3	0.0023	0.0477	-0.0946
Triple 4	0.0710	0.05067	0.1045
Triple 5	0.0388	0.04312	-0.0085

Top 5 most frequent relations:
[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasReference', 709),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/term', 468),
('http://www.w3.org/1999/02/22-rdf-syntax-ns#type', 275),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/level', 48)]

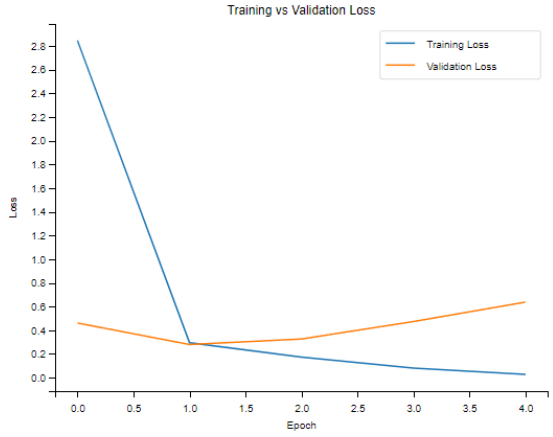
Relation with highest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/level', 0.29218049774256843)

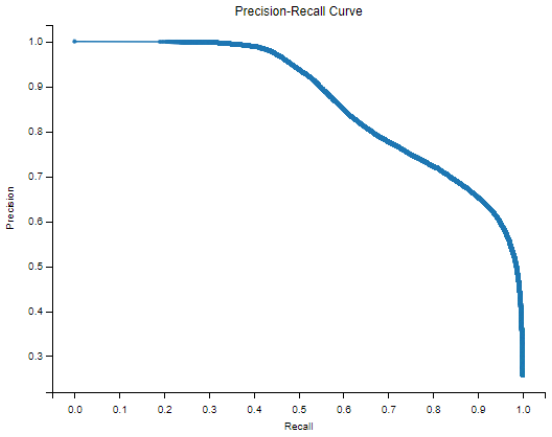
Relation with lowest average probability:
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasReference',
0.2641741103671323)

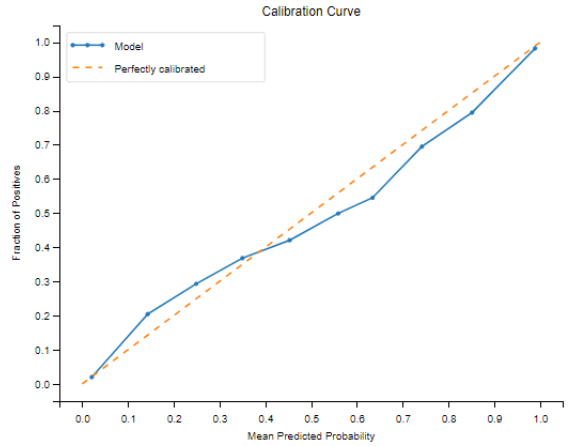
XX. Experiment 10

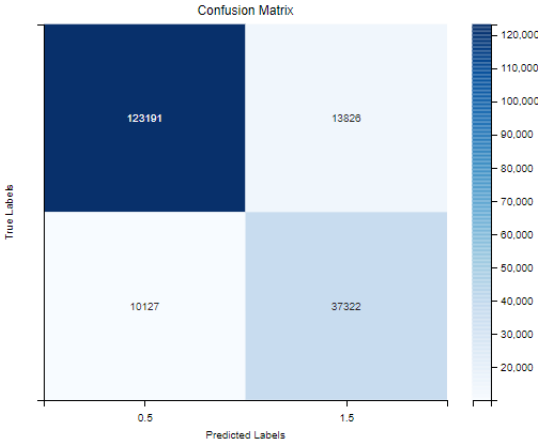
	Mean Silhouette	Mean Davies-Bouldin	Mean Calinski-Harabasz
	0.8086	0.2453	4373.2786

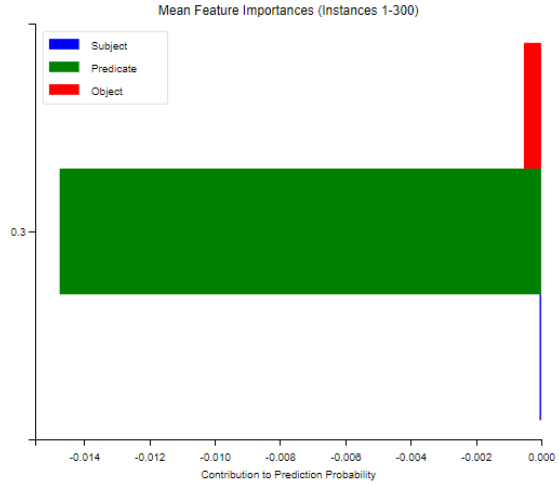
C1	C2	C3	C4	C5	C6	C7	C8
30	76	0	0	0.2180	0.8965	0.0004	0.2173

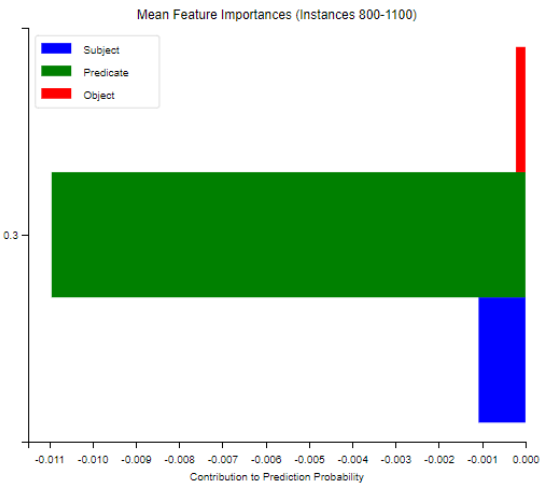

Training vs Validation Loss
Training Loss
Validation Loss
Loss
Epoch

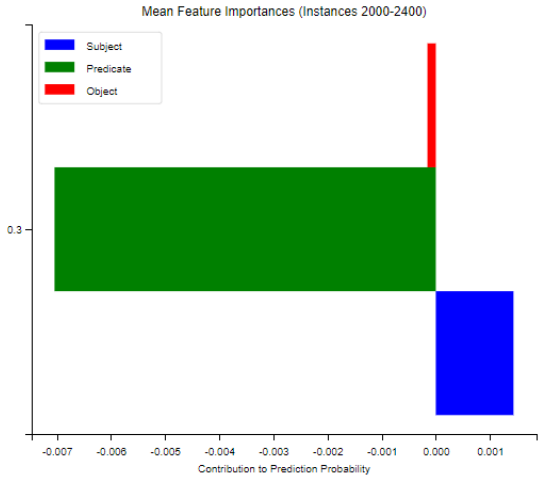

Precision-Recall Curve
Precision
Recall


Calibration Curve
Model
Perfectly calibrated
Fraction of Positives
Mean Predicted Probability


Confusion Matrix
123191
13826
10127
37322
True Labels
Predicted Labels


Mean Feature Importances (Instances 1-300)
Subject
Predicate
Object
Contribution to Prediction Probability


Mean Feature Importances (Instances 800-1100)
Subject
Predicate
Object
Contribution to Prediction Probability



	subject	predicate	object
Triple 1	-0.0429	-0.05529	0.0018
Triple 2	-0.0647	-0.01587	-0.1636
Triple 3	0.0211	-0.00969	0.0477
Triple 4	0.0961	-0.00157	0.0520
Triple 5	0.0199	0.04658	0.0747

Top 5 most frequent relations:

[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasFrequentTerm', 256),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/keyword', 207),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/id', 204),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasCode', 137),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasSubTheme', 116)]

Relations appearing only once:
['https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasCategoryOfGlossaryArticle',
'http://www.w3.org/2002/07/owl#unionOf',
'http://www.w3.org/2002/07/owl#versionInfo',
'https://ec.europa.eu/eurostat/NLP4StatRef/ontology/relatedTerm',
'https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasTopic',
'http://www.w3.org/2002/07/owl#backwardCompatibleWith']

Relation with highest average probability:

('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasTopic', 0.5259029865264893)

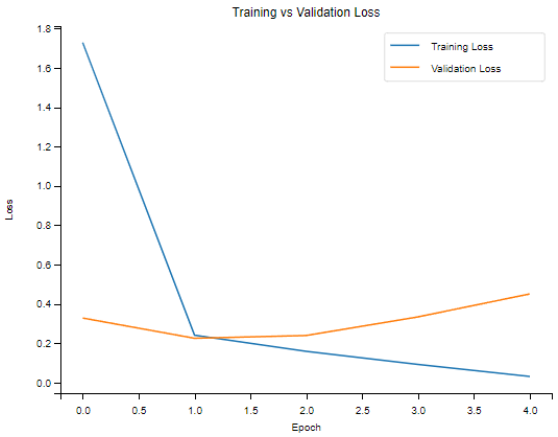
Relation with lowest average probability:

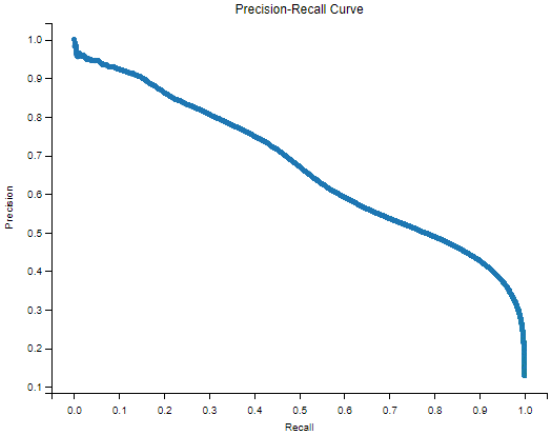
('http://www.w3.org/2002/07/owl#unionOf', 0.015426167286932468)

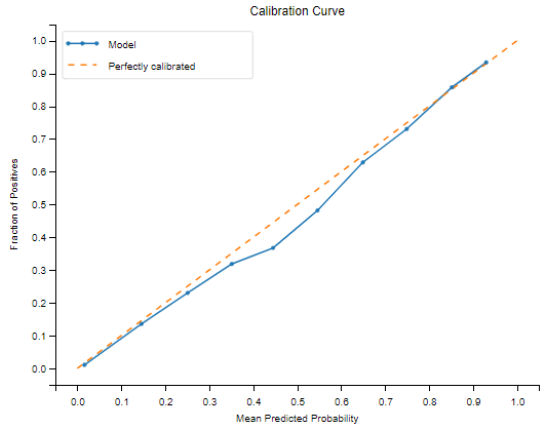
XXI. Experiment 11

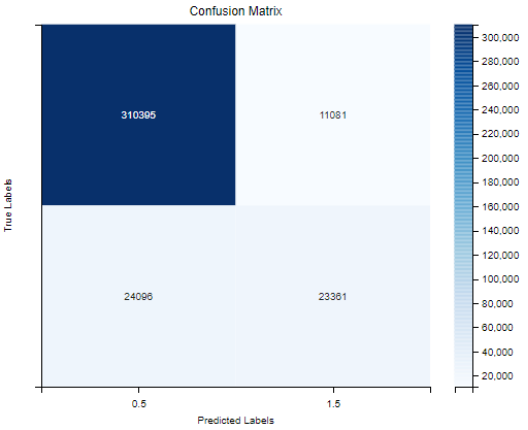
	Mean Silhouette	Mean Davies-Bouldin	Mean Calinski-Harabasz
	0.8097	0.2443	4150.0788

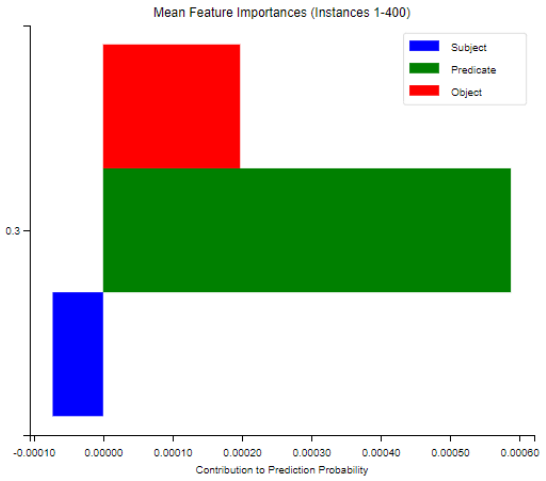
C1	C2	C3	C4	C5	C6	C7	C8
21	3	0	0	0.0425	0.7365	0.0003	0.0872

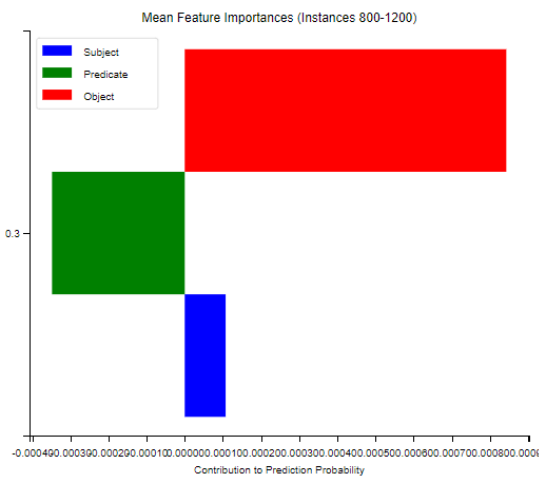


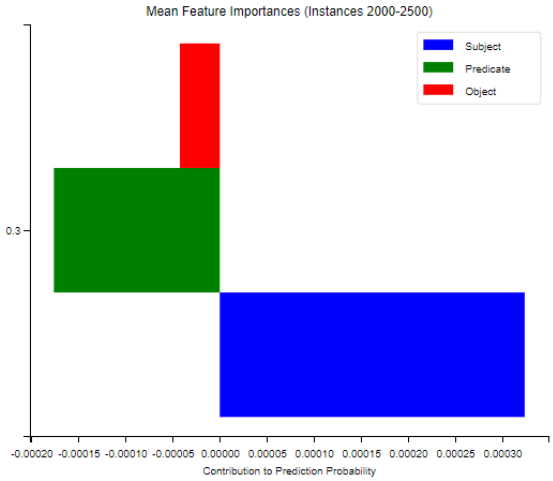












	subject	predicate	object
Triple 1	-0.0207	0.023996	-0.0123
Triple 2	-0.0183	-0.01819	0.0149
Triple 3	-0.0101	-0.01473	-0.0079
Triple 4	0.0381	-0.02072	0.0159
Triple 5	-0.0060	0.02516	-0.0064

Top 5 most frequent relations:

[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/remark', 612),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasCode', 170),
('http://www.w3.org/2000/01/rdf-schema#subClassOf', 129),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasURI', 125),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/title', 119)]

Relations appearing only once:

['http://www.w3.org/2000/01/rdf-schema#label',
'https://ec.europa.eu/eurostat/NLP4StatRef/ontology/fileLink',

'http://www.w3.org/1999/02/22-rdf-syntax-ns#type',

'https://ec.europa.eu/eurostat/NLP4StatRef/ontology/relatedTheme']

Relation with highest average probability:

('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasParagraph',
0.1194011508487165)

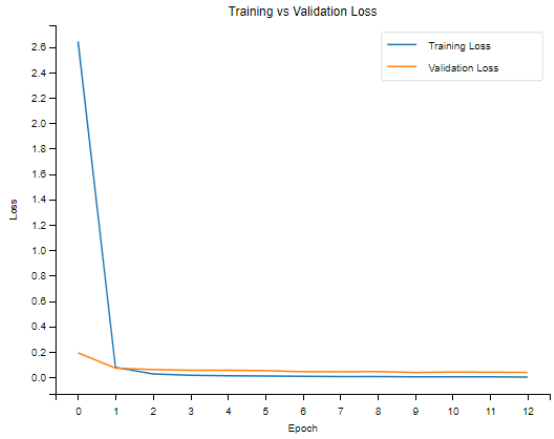
Relation with lowest average probability:

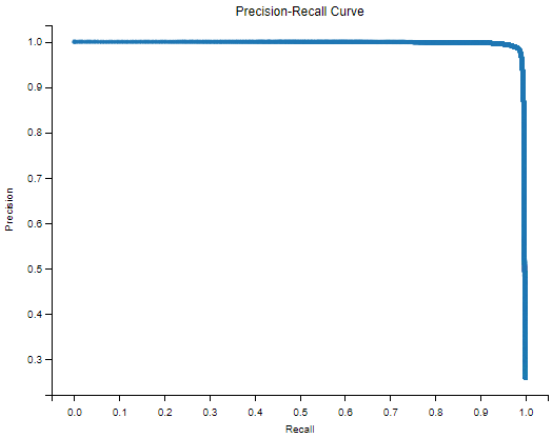
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/relatedTheme',
0.002175799338147044)

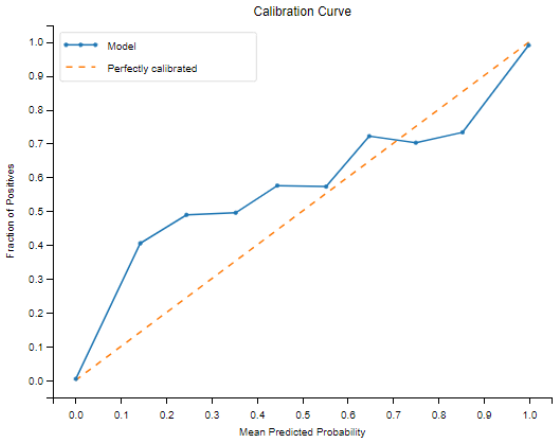
XXII. Experiment 12

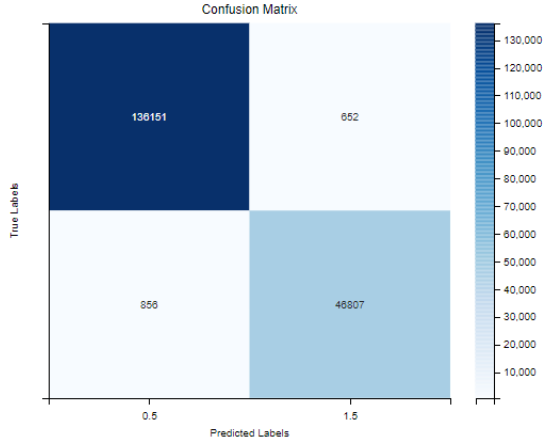
	Mean Silhouette	Mean Davies-Bouldin	Mean Calinski-Harabasz
	0.8049	0.2518	4022.4616

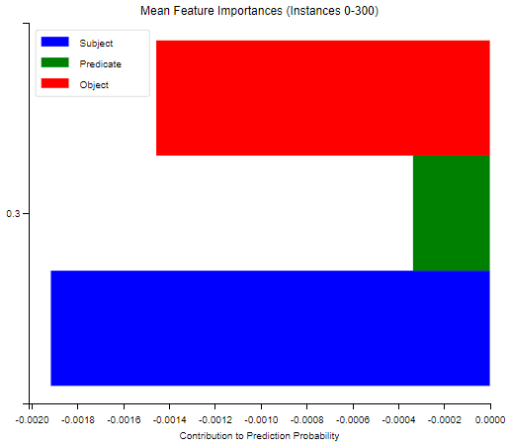
C1	C2	C3	C4	C5	C6	C7	C8
20	413	356	17	0.3006	1.0000	0.0001	0.4236

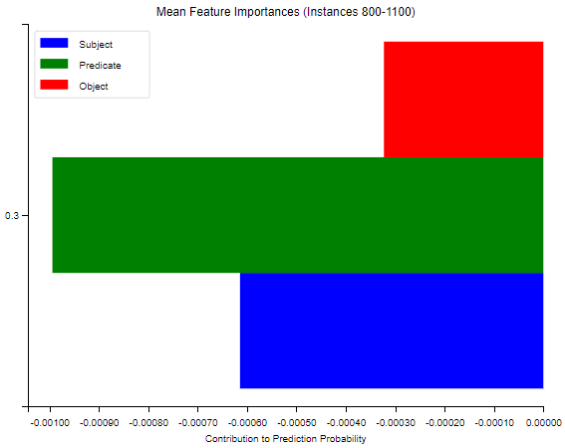


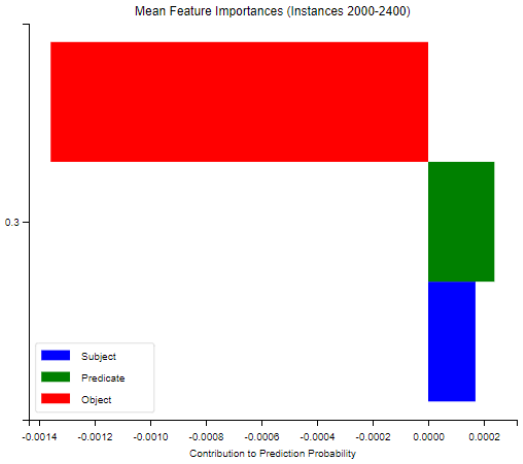












	subject	predicate	object
Triple 1	0.2565	0.10448	0.3953
Triple 2	-0.2190	0.07574	0.2227
Triple 3	-0.0508	-0.05518	-0.6714
Triple 4	0.2774	0.09223	0.3310
Triple 5	-0.1902	-0.1351	-0.6626

Top 5 most frequent relations:

[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasURI', 319),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/definition', 301),
('http://www.w3.org/2000/01/rdf-schema#subClassOf', 189),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasCode', 167),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/relatedTerm', 143)]

Relations appearing only once:
['https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasOECDTheme',
'https://ec.europa.eu/eurostat/NLP4StatRef/ontology/fileLink']

Relation with highest average probability:

('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasParagraph',
0.5013474763836712)

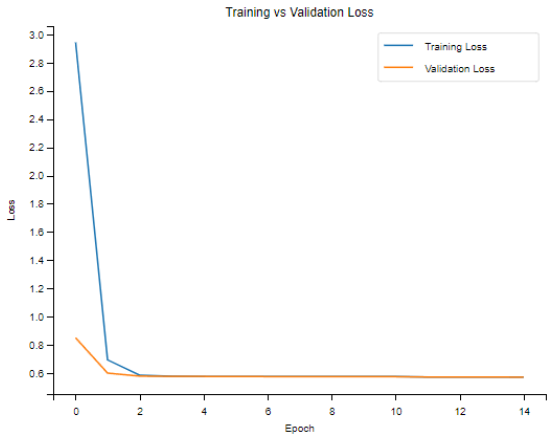
Relation with lowest average probability:

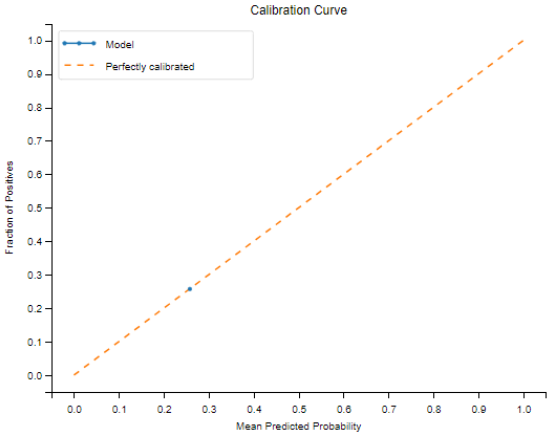
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/fileLink', 0.0003957330482080579)

XXIII. Experiment 13

	Mean Silhouette	Mean Davies-Bouldin	Mean Calinski-Harabasz
	0.7835	0.2856	3054.1802

	C1	C2	C3	C4	C5	C6	C7	C8
	1	0	0	0	0.2580	0.2580	0.2580	0.0000





Top 5 most frequent relations:

[('http://www.w3.org/2000/01/rdf-schema#subPropertyOf', 1500)]

Relation with highest average probability:

('http://www.w3.org/2000/01/rdf-schema#subPropertyOf', 0.2579798400402069)

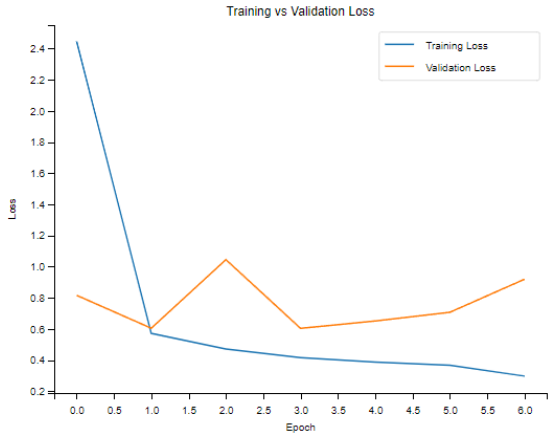
Relation with lowest average probability:

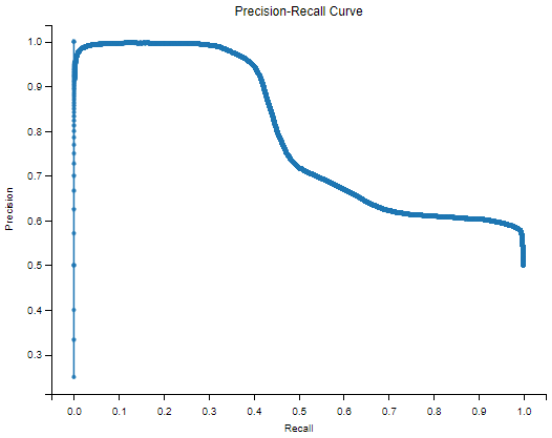
('http://www.w3.org/2000/01/rdf-schema#subPropertyOf', 0.2579798400402069)

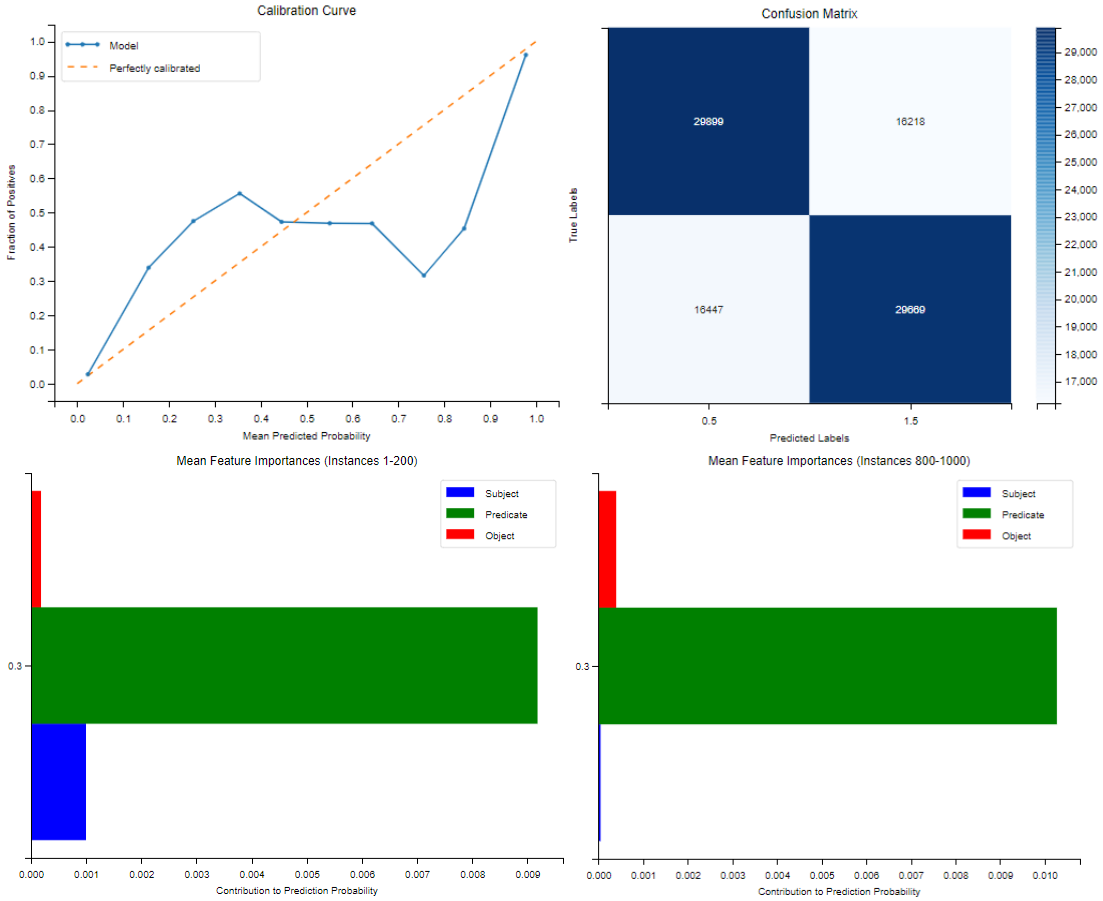
XXIV. Experiment 14

	Mean Silhouette	Mean Davies-Bouldin	Mean Calinski-Harabasz
	0.8125	0.2416	3715.9306

	C1	C2	C3	C4	C5	C6	C7	C8
	8	386	0	0	0.4228	0.8762	0.0014	0.2588







	subject	predicate	object
Triple 1	0.01115	0.0160	-0.24187
Triple 2	-0.01493	0.2559	-0.07520
Triple 3	-0.00351	-0.41166	-0.07640
Triple 4	0.01936	0.25983	-0.03920
Triple 5	-0.01073	0.03038	0.42009

Top 5 most frequent relations:

[('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/databasePath', 1124),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasFrequentTerm', 350),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasCode', 14),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/content', 4),
('https://ec.europa.eu/eurostat/NLP4StatRef/ontology/hasParagraph', 3)]

Relations appearing only once:

['http://www.w3.org/2000/01/rdf-schema#label']

Relation with highest average probability:

('http://www.w3.org/2002/07/owl#equivalentClass', 0.6497162580490112)

Relation with lowest average probability:

('http://www.w3.org/2000/01/rdf-schema#label', 0.009083980694413185)